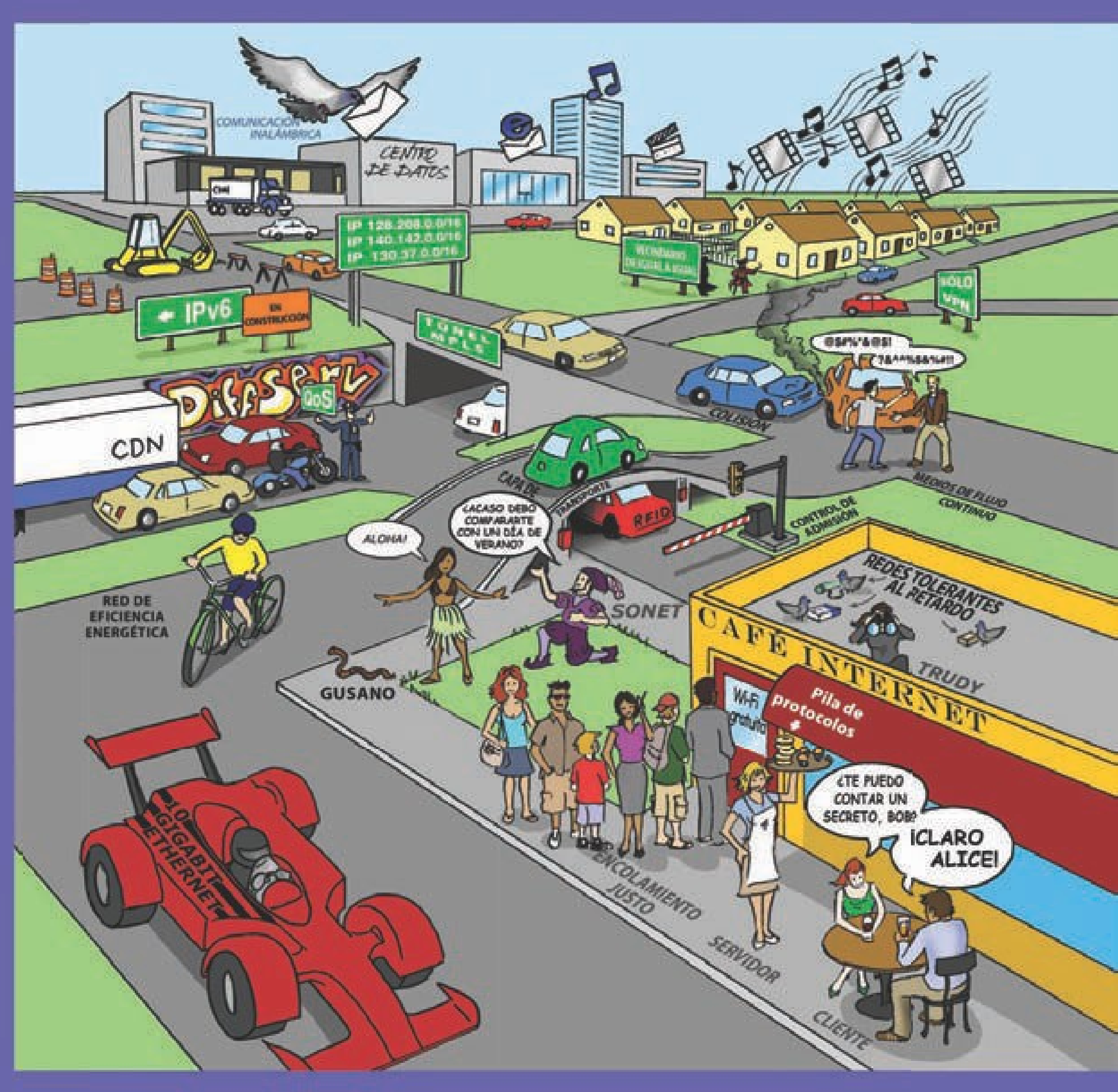
REDES DE COMPUTADORAS

QUINTA EDICIÓN

TANENBAUM | WETHERALL



ALWAYS LEARNING

PEARSON

Redes
de computadoras

Quinta edición

Redes de computadoras

Quinta edición

Andrew S. Tanenbaum
Vrije Universiteit
Amsterdam

David J. Wetherall
University of Washington
Seatle, Washington

TRADUCCIÓN

Alfonso Vidal Romero Elizondo
Ingeniero en Sistemas Electrónicos
Instituto Tecnológico y de Estudios
Superiores de Monterrey-Campus Monterrey

REVISIÓN TÉCNICA

M. en C. Cyntia E. Enríquez Ortiz
Escuela Superior de Cómputo
Instituto Politécnico Nacional

PEARSON

Datos de catalogación bibliográfica

ANDREW S. TANENBAUM y DAVID J. WETHERALL

Redes de computadoras
Quinta edición

PEARSON EDUCACIÓN, México, 2012

ISBN: 978-607-32-0817-8
Área: Computación

Formato: 20 × 25.5 cm Páginas: 816

ISBN 9780132126953

Edición en español

Editor:	Luis M. Cruz Castillo e-mail: luis.cruz@pearson.com
Editor de desarrollo:	Bernardino Gutiérrez Hernández
Supervisor de producción:	Juan José García Guzmán

QUINTA EDICIÓN, 2012

Cámara Nacional de la Industria Editorial Mexicana. Reg. núm. 1031.

ISBN VERSIÓN IMPRESA: 978-607-32-0817-8
ISBN VERSIÓN E-BOOK: 978-607-32-0818-5
ISBN E-CHAPTER: 978-607-32-0819-2

Impreso en México. Printed in Mexico.
1 2 3 4 5 6 7 8 9 0 - 14 13 12 11



www.pearsoneducacion.net

Para Suzanne, Barbara, Daniel, Aron, Marvin, Matilde
y a la memoria de Bram y Sweetie
(AST)

Para Katrim, Lucy y Pepper
(DJW)

CONTENIDO

Prefacio

Este libro se encuentra ahora en su quinta edición. Cada edición ha correspondido a una fase distinta en cuanto a la forma en que se utilizaban las redes de computadoras. Cuando apareció la primera edición en 1980, las redes eran una curiosidad académica. Para la segunda edición, en 1988, las redes se utilizaban en las universidades y en las grandes empresas. Cuando salió al mercado la tercera edición, en 1996, las redes de computadoras (en especial Internet) se habían convertido en una realidad diaria para millones de personas. Ya para la cuarta edición, en 2003, las redes inalámbricas y las computadoras móviles se habían vuelto herramientas comunes para acceder a la web e Internet. Ahora, en la quinta edición, las redes tratan sobre la distribución de contenido (en especial los videos que utilizan CDN y redes de igual a igual) y los teléfonos móviles son pequeñas computadoras con Internet.

Novedades de la quinta edición

Entre los diversos cambios que se presentan en este libro, el más importante es la incorporación del profesor David J. Wetherall como coautor. El profesor Wetherall posee una extensa experiencia con las redes, tiene más de 20 años experimentando con las redes de área metropolitana. Desde entonces ha trabajado con las redes inalámbricas e Internet, además de fungir como profesor en la University of Washington, en donde ha enseñado y realizado investigaciones sobre las redes de computadoras y temas relacionados durante la última década.

Desde luego, el libro también incluye cambios sustanciales para estar a la par con el siempre cambiante mundo de las redes computacionales. Algunos de estos cambios incluyen material actualizado y nuevo sobre:

- Redes inalámbricas (802.12 y 802.16).
- Las redes 3G que utilizan los teléfonos inteligentes.

- Redes RFID y de sensores.
- Distribución de contenido mediante el uso de CDN.
- Redes de igual a igual.
- Medios en tiempo real (de fuentes almacenadas, de flujo continuo y en vivo).
- Telefonía por Internet (voz sobre IP).
- Redes tolerantes al retraso.

A continuación encontrará una descripción más detallada por capítulo.

El capítulo 1 tiene la misma función de presentación que en la cuarta edición, pero revisamos y actualizamos el contenido. Aquí hablamos sobre Internet, las redes de teléfonos móviles, 802.11, las redes RFID y de sensores como ejemplos de redes computacionales. Eliminamos el material sobre la Ethernet original (con sus conexiones tipo vampiro), junto con el material sobre ATM.

El capítulo 2, que trata sobre la capa física, cuenta con una cobertura más amplia de la modulación digital (incluyendo la multiplexación OFDM y su popularidad en las redes inalámbricas) y las redes 3G (basadas en CDMA). También se describen las nuevas tecnologías, incluyendo *Fiber to Home* o FTTH (fibra hasta el hogar) y las redes a través del cableado eléctrico.

El capítulo 3, que trata sobre los enlaces punto a punto, se mejoró de dos formas. Se actualizó el material sobre los códigos para detección y corrección de errores, además de incluir una breve descripción de los códigos modernos importantes en la práctica (por ejemplo, los códigos convolucional y LDPC). Los ejemplos de protocolos utilizan ahora un paquete sobre SONET y ADSL. Tuvimos que eliminar el material sobre verificación de protocolos debido a que en la actualidad no se utiliza mucho.

En el capítulo 4, sobre la subcapa MAC, los principios son eternos pero las tecnologías han cambiado. Se rediseñaron las secciones sobre las redes de ejemplo de manera acorde, incluyendo las redes Gigabit Ethernet, 802.11, 802.16, Bluetooth y RFID. También se actualizó la información sobre las redes LAN conmutadas, incluyendo las redes VLAN.

El capítulo 5, que trata sobre la capa de red, cubre los mismos conceptos que en la cuarta edición. Se hicieron revisiones para actualizar el material y agregar más detalles, en especial sobre la calidad del servicio (lo cual es relevante para los medios en tiempo real) y la interconectividad (*internetworking*). Se expandieron las secciones sobre BGP, OSPF y CIDR, así como el material sobre el enrutamiento multidifusión (*multicast*). Ahora se incluye también el enrutamiento *anycast*.

En el capítulo 6, sobre la capa de transporte, se agregó, modificó y eliminó material. El nuevo material describe las redes tolerantes al retardo y el control de congestión en general. El material modificado actualiza y expande la cobertura sobre el control de congestión en TCP. El material eliminado describe las capas de red orientadas a la conexión, algo que se ve rara vez en la actualidad.

En el capítulo 7, que trata sobre aplicaciones, también se actualizó la información y se aumentó el contenido. Aunque el material sobre DNS y correo electrónico es similar al de la cuarta edición, en los últimos años se han suscitado varios acontecimientos en cuanto al uso de web, los medios de flujo continuo y la distribución de contenido. Asimismo, se actualizaron las secciones sobre web y los medios de flujo continuo. Hay una nueva sección sobre la distribución de contenido, incluyendo las redes CDN y de igual a igual.

El capítulo 8, sobre la seguridad, trata aún la criptografía tanto simétrica como de clave pública para la confidencialidad y autenticidad. Se actualizó el material sobre las técnicas utilizadas en la práctica, incluyendo *firewalls* y redes VPN; además se agregó material nuevo sobre la seguridad en redes 802.11 y Kerberos V5.

El capítulo 9 contiene una lista renovada de lecturas sugeridas y una extensa bibliografía con más de 300 citas de literatura actual. Más de la mitad de éstas son de artículos y libros, mientras que el resto son citas de artículos clásicos.

Lista de siglas y acrónimos

Los libros de computación están llenos de acrónimos y éste no es la excepción. Para cuando termine de leerlo, los siguientes acrónimos le serán familiares: ADSL, AES, AJAX, AODV, AP, ARP, ARQ, AS, BGP, BOC, CDMA, CDN, CGI, CIDR, CRL, CSMA, CSS, DCT, DES, DHCP, DHT, DIFS, DMCA, DMT, DMZ, DNS, DOCSIS, DOM, DSLAM, DTN, FCFS, FDD, FDDI, FDM, FEC, FIFO, FSK, FTP, GPRS, GSM, HDTV, HFC, HMAC, HTTP, IAB, ICANN, ICMP, IDEA, IETF, IMAP, IMP, IP, IPTV, IRTF, ISO, ISP, ITU, JPEG, JSP, JVM, LAN, LATA, LEC, LEO, LLC, LSR, LTE, MAN, MFJ, MIME, MPEG, MPLS, MSC, MTSO, MTU, NAP, NAT, NRZ, NSAP, OFDM, OSI, OSPF, PAWS, PCM, PGP, PIM, PKI, POP, POTS, PPP, PSTN, QAM, QPSK, RED, RFC, RFID, RPC, RSA, RTSP, SHA, SIP, SMTP, SNR, SOAP, SONET, SPE, SSL, TCP, TDD, TDM, TSAP, UDP, UMTS, URL, VLAN, VSAT, WAN, WDM y XML. No se preocupe; cada uno aparecerá en **negritas** y lo definiremos con detalle antes de usarlo. Hagamos una prueba divertida: revise cuántas de estas siglas puede identificar *antes* de leer el libro, escriba el número al margen y vuelva a intentarlo *después* de leer el libro.

Cómo usar este libro

Para ayudar a los profesores a utilizar este libro como texto para cursos cuya duración puede variar entre un trimestre o un semestre, estructuramos los subtítulos de los capítulos como: material básico y opcional. Las secciones marcadas con un asterisco (*) en el contenido son *opcionales*. Si hay una sección importante marcada de esta forma (vea por ejemplo la sección 2.7), entonces todas sus subsecciones son opcionales, ya que proveen material sobre tecnologías de redes que es útil pero se puede omitir en un curso corto sin perder la continuidad. Desde luego que hay que animar a los estudiantes a que lean también esas secciones, siempre y cuando tengan tiempo suficiente, ya que todo el material está actualizado y es valioso.

Materiales didácticos para los profesores

Los siguientes materiales didácticos (en inglés) "protegidos para los profesores" están disponibles en *www.pearsoneducacion.net/tanenbaum*, sitio web creado para este libro. Para obtener un nombre de usuario y contraseña, póngase en contacto con su representante local de Pearson.

- Manual de soluciones.
- Diapositivas en PowerPoint.

Materiales didácticos para los estudiantes

Los recursos para los estudiantes (en inglés) están disponibles también en el sitio web de este libro: *www.pearsoneducacion.net/tanenbaum*, a través del vínculo *Companion Website*, e incluyen:

- Recursos web.
- Figuras, tablas y programas del libro.
- Demostración de esteganografía.

AGRADECIMIENTOS

Muchas personas nos ayudaron durante el desarrollo de esta quinta edición. Nos gustaría agradecer en especial a Emmanuel Agu (Worcester Polytechnic Institute), Yoris Au (University of Texas en San Antonio), Nikhil Bhargava (Aircom International, Inc.), Michael Buettner (University of Washington), John Day (Boston University), Kevin Fall (Intel Labs), Ronald Fulle (Rochester Institute of Technology), Ben Greenstein (Intel Labs), Daniel Halperin (University of Washington), Bob Kinicki (Worcester Polytechnic Institute), Tadayoshi Kohno (University of Washington), Sarvish Kulkarni (Villanova University), Hank Levy (University of Washington), Ratul Mahajan (Microsoft Research), Craig Partridge (BBN), Michael Piatek (University of Washington), Joshua Smith (Intel Labs), Neil Spring (University of Maryland), David Teneyuca (University of Texas en San Antonio), Tammy VanDegrift (University of Portland) y Bo Yuan (Rochester Polytechnic Institute), por aportar ideas y retroalimentación. Melody Kadenko y Julie Svendsen brindaron apoyo administrativo al profesor Wetherall.

Shivakant Mishra (University of Colorado en Boulder) y Paul Nagin (Chimborazo Publishing, Inc) idearon muchos de los problemas nuevos y retadores de fin de capítulo. Tracy Dunkelberger, nuestra editora en Pearson, fue tan útil como siempre en muchas tareas tanto grandes como pequeñas. Melinda Haggerty y Jeff Holcomb hicieron un excelente trabajo al cuidar que todo se llevara a cabo sin problemas. Steve Armstrong (LeTourneau University) preparó las diapositivas de PowerPoint. Stephen Turner (University of Michigan en Flint) revisó meticulosamente los recursos web y los simuladores que acompañan el libro. Rachel Head, nuestra correctora de estilo, es una híbrido fuera de lo común: tiene la vista de águila y la memoria de un elefante. Después de leer sus correcciones, ambos autores nos preguntamos cómo fue posible que pasáramos del tercer grado de primaria.

Por último, llegamos a las personas más importantes. Suzanne ha pasado por esto 19 veces hasta ahora y aún sigue con su paciencia y amor interminables. Barbara y Marvin ahora conocen la diferencia entre los buenos libros de texto y los malos libros que siempre son una inspiración para producir buenos. Daniel y Matilde son los miembros más recientes de nuestra familia, a

quienes recibimos con gran afecto. Es poco probable que Aron vaya a leer pronto este libro, pero de todas formas le gustan las bonitas imágenes de la figura 8-54. Katrin y Lucy brindaron su apoyo incondicional y siempre lograron mantener una sonrisa en mi rostro. Gracias.

ANDREW S. TANENBAUM
DAVID J. WETHERALL

ACERCA DE LOS AUTORES

Andrew S. Tanenbaum tiene una licenciatura en Ciencias (S.B.) por el MIT y un doctorado por la University of California, en Berkeley. Actualmente es profesor de Ciencias Computacionales en la Vrije Universiteit, en donde ha enseñado durante más de 30 años sobre sistemas operativos, redes y temas relacionados. Su investigación actual es sobre los sistemas operativos muy confiables, aunque ha trabajado también en compiladores, sistemas distribuidos, segu-ridad y otros temas. Estos proyectos de investigación han generado más de 150 artículos de referencia en publicaciones especializadas e infinidad de conferencias.

El profesor Tanenbaum también ha sido coautor de cinco libros que a la fecha han aparecido en 19 ediciones. Sus libros se han traducido a 21 idiomas, que varían desde el vasco hasta el tailandés y se utilizan en universidades de todo el mundo; en total, hay 159 versiones (combinadas idiomas y ediciones) que se listan en *www.cs.vu.nl/~ast/publications*.

El profesor Tanenbaum también ha producido un volumen considerable de software, incluyendo el paquete de compilador Amsterdam (un compilador portátil reorientable), Amoeba (uno de los primeros sistemas distribuidos utilizados en LAN) y Globe (un sistema distribuido de área amplia).

También es autor de MINIX, un pequeño clon de UNIX que en un principio estaba destinado a usarse en los laboratorios de programación estudiantiles. Fue la inspiración directa de Linux y la plataforma en la que se desarrolló inicialmente este sistema operativo. La versión actual de MINIX, conocida como MINIX3, ahora se enfoca en ser un sistema operativo en extremo confiable y seguro. El profesor Tanenbaum considerará su trabajo terminado cuando las computadoras no estén equipadas con un botón de reinicio y ningún ser humano haya experimentado una falla del sistema. MINIX3 es un proyecto continuo de código fuente abierto, al cual usted está invitado a contribuir. Vaya a *www.minix3.org* para que descargue una copia gratuita y averigüe las novedades.

Tanenbaum es miembro del ACM y del IEEE, además de pertenecer a la Real Academia de Artes y Ciencias de los Países Bajos. También ha ganado numerosos premios científicos, como:

- Premio TAA McGuffey, en 2010, para libros de ciencias computacionales e ingeniería.
- Medalla James H. Mulligan Jr., del IEEE, en 2007, por sus contribuciones a la educación.
- Premio TAA Texty, en 2002, para libros de ciencias computacionales e ingeniería.
- Premio ACM/SIGCSE, en 1997, por sus sorprendentes contribuciones a la educación en las ciencias computacionales.
- Premio Karl V. Karlstrom, del ACM, en 1994, para educadores sobresalientes.

Puede visitar su página de World Wide Web en *http://www.cs.vu.nl/~ast/*.

David J. Wetherall es profesor asociado de Ciencias Computacionales e Ingeniería en la University of Washington, en Seattle, además de trabajar como consultor para Intel Labs, en Seattle. Es originario de Australia, en donde recibió su licenciatura en ingeniería eléctrica por la University of Western Australia, y su doctorado en Ciencias Computacionales del MIT.

El profesor Wetherall ha trabajado en el área de las redes durante las últimas dos décadas. Su investigación se enfoca en los sistemas de red, en especial las redes inalámbricas y la computación móvil, el diseño de protocolos de Internet y la medición en las redes.

Recibió el premio ACM SIGCOMM Test-of-Time por su investigación pionera en las redes activas, una arquitectura para introducir los nuevos servicios de red con rapidez. Recibió el premio William Bennet, del IEEE, por sus descubrimientos en el mapeo de Internet. Su investigación se reconoció con un premio NSF CAREER, en 2002, y fue becario Sloan, en 2004.

Además de impartir clases sobre redes, el profesor Wetherall participa en la comunidad de investigación sobre redes. Ha codirigido los comités de programas de SIGCOMM, NSDI y MobiSys, además de ser cofundador de los talleres de trabajo HotNets, de la ACM. Ha participado en numerosos comités de programas para conferencias sobre redes y es editor de la publicación *Computer Communication Review*, de la ACM.

Puede visitar su página de World Wide Web en *http://djw.cs.washington.edu*.

1

INTRODUCCIÓN

Cada uno de los tres últimos siglos ha estado dominado por una nueva tecnología. El siglo XVIII fue la época de los grandes sistemas mecánicos que dieron paso a la Revolución Industrial. El siglo XIX fue la era de la máquina de vapor. Durante el siglo XX, la tecnología clave fue la recopilación, procesamiento y distribución de información. Entre otros desarrollos vimos la instalación de las redes telefónicas a nivel mundial, la invención de la radio y la televisión, el nacimiento y crecimiento sin precedentes de la industria de la computación, el lanzamiento de satélites de comunicaciones y, desde luego, Internet.

Como resultado del vertiginoso progreso tecnológico, estas áreas están convergiendo con rapidez en el siglo XXI, y las diferencias entre recolectar, transportar, almacenar y procesar información están desapareciendo rápidamente. Las organizaciones con cientos de oficinas esparcidas sobre una amplia área geográfica dan por sentado como algo rutinario la capacidad de examinar el estado actual, aun de su oficina más remota, con sólo presionar un botón. A medida que aumenta nuestra habilidad para recopilar, procesar y distribuir la información, la demanda por un procesamiento aún más complejo de la información aumenta rápidamente.

A pesar de que la industria de la computación es joven si se le compara con otras (como la automotriz y la de transporte aéreo), las computadoras han progresado de manera espectacular en un periodo muy corto. Durante las primeras dos décadas de su existencia, estos sistemas estaban altamente centralizados y por lo general se encontraban dentro de un salón grande e independiente. Era común que este salón tuviera paredes de vidrio, a través de las cuales los visitantes podían mirar boquiabiertos la gran maravilla electrónica que había en su interior. Una empresa o universidad de tamaño mediano apenas lograba tener una o dos computadoras, mientras que las instituciones muy grandes tenían, cuando mucho, unas cuantas docenas. La idea de que en un lapso de 40 años se produjeran en masa miles de millones de computadoras mucho más poderosas y del tamaño de una estampilla postal era en ese entonces mera ciencia ficción.

La fusión de las computadoras y las comunicaciones ha tenido una profunda influencia en cuanto a la manera en que se organizan los sistemas de cómputo. El concepto una vez dominante del "centro de cómputo" como un salón con una gran computadora a la que los usuarios llevaban su trabajo para procesarlo es ahora totalmente obsoleto (aunque los centros de datos que contienen miles de servidores de Internet se están volviendo comunes). El viejo modelo de una sola computadora para atender todas las necesidades computacionales de la organización se ha reemplazado por uno en el que un gran número de computadoras separadas pero interconectadas realizan el trabajo. A estos sistemas se les conoce como **redes de computadoras**. El diseño y la organización de estas redes es el objetivo de este libro.

A lo largo del libro utilizaremos el término "red de computadoras" para referirnos a un conjunto de computadoras autónomas interconectadas mediante una sola tecnología. Se dice que dos computadoras están interconectadas si pueden intercambiar información. La conexión no necesita ser a través de un cable de cobre; también se puede utilizar fibra óptica, microondas, infrarrojos y satélites de comunicaciones. Las redes pueden ser de muchos tamaños, figuras y formas, como veremos más adelante. Por lo general se conectan entre sí para formar redes más grandes, en donde **Internet** es el ejemplo más popular de una red de redes.

Existe una gran confusión en la literatura entre una red de computadoras y un **sistema distribuido**. La diferencia clave está en que en un sistema distribuido, un conjunto de computadoras independientes aparece frente a sus usuarios como un solo sistema coherente. Por lo general, tiene un modelo o paradigma único que se presenta a los usuarios. A menudo se utiliza una capa de software encima del sistema operativo, conocido como **middleware**; esta capa es responsable de implementar este modelo. Un ejemplo reconocido de un sistema distribuido es la **World Wide Web**. Este sistema opera sobre Internet y presenta un modelo en el cual todo se ve como un documento (página web).

En una red de computadoras no existe esta coherencia, modelo ni software. Los usuarios quedan expuestos a las máquinas reales, sin que el sistema haga algún intento por hacer que éstas se vean y actúen de una manera coherente. Si las máquinas tienen distinto hardware y distintos sistemas operativos, es algo que está a la vista de los usuarios. Si un usuario desea ejecutar un programa en un equipo remoto, tiene que iniciar sesión en esa máquina y ejecutarlo ahí.

En efecto, un sistema distribuido es un sistema de software construido sobre una red. El software le ofrece un alto nivel de cohesión y transparencia. Por ende, la distinción entre una red y un sistema distribuido recae en el software (en especial, el sistema operativo) y no en el hardware.

Sin embargo, los dos temas se superponen de manera considerable. Por ejemplo, tanto los sistemas distribuidos como las redes de computadoras necesitan mover archivos. La diferencia recae en quién invoca el movimiento, si el sistema o el usuario. Aunque este libro se enfoca principalmente en las redes, muchos de los temas también son importantes en los sistemas distribuidos. Para obtener más información, vea Tanenbaum y Van Steen (2007).

1.1 USOS DE LAS REDES DE COMPUTADORAS

Antes de examinar las cuestiones técnicas con detalle, vale la pena dedicar cierto tiempo a señalar por qué las personas están interesadas en las redes de computadoras y para qué se pueden utilizar. Después de todo, si nadie estuviera interesado en ellas, se construirían muy pocas. Empezaremos con las cuestiones tradicionales en las empresas, después pasaremos a las redes domésticas y a los acontecimientos recientes en relación con los usuarios móviles, para terminar con las cuestiones sociales.

1.1.1 Aplicaciones de negocios

La mayoría de las empresas tienen una cantidad considerable de computadoras. Por ejemplo, tal vez una empresa tenga una computadora para cada empleado y las utilice para diseñar productos, escribir folletos y llevar la nómina. Al principio, algunas de estas computadoras tal vez hayan trabajado aisladas unas de otras, pero en algún momento, la administración podría decidir que es necesario conectarlas para distribuir la información en toda la empresa.

En términos generales, el asunto es **compartir recursos** y la meta es que todos los programas, equipo y en especial los datos estén disponibles para cualquier persona en la red, sin importar la ubicación física del recurso o del usuario. Un ejemplo obvio y de uso popular es el de un grupo de empleados de oficina que comparten una impresora. Ninguno de los individuos necesita realmente una impresora privada, por otro lado, una impresora en red de alto volumen es más económica, veloz y fácil de mantener que una extensa colección de impresoras individuales.

Pero, probablemente, compartir información sea aún más importante que compartir recursos físicos como impresoras y sistemas de respaldo en cinta magnética. Las empresas tanto pequeñas como grandes dependen vitalmente de la información computarizada. La mayoría tiene registros de clientes, información de productos, inventarios, estados de cuenta, información fiscal y muchos datos más en línea. Si de repente todas sus computadoras se desconectaran de la red, un banco no podría durar más de cinco minutos. Una planta moderna de manufactura con una línea de ensamble controlada por computadora no duraría ni cinco segundos. Incluso una pequeña agencia de viajes o un despacho legal compuesto de tres personas son altamente dependientes de las redes de computadoras para permitir a los empleados acceder a la información y los documentos relevantes de manera instantánea.

En las empresas más pequeñas es probable que todas las computadoras se encuentren en una sola oficina o tal vez en un solo edificio, pero en las empresas más grandes las computadoras y empleados se encuentran esparcidos en docenas de oficinas y plantas en muchos países. Sin embargo, un vendedor en Nueva York podría requerir acceso a una base de datos que se encuentra en Singapur. Las redes conocidas como **VPN** (**Redes Privadas Virtuales**, del inglés *Virtual Private Networks*) se pueden usar para unir las redes individuales, ubicadas en distintos sitios, en una sola red extendida. En otras palabras, el simple hecho de que un usuario esté a 15 000 km de distancia de sus datos no debe ser impedimento para que los utilice como si fueran locales. Podemos sintetizar este objetivo al decir que es un intento por acabar con la “tiranía de la geografía”.

En términos más simples, imaginemos el sistema de información de una empresa como si estuviera constituido por una o más bases de datos con información de la empresa y cierto número de empleados que necesitan acceder a esos datos en forma remota. En este modelo, los datos se almacenan en poderosas computadoras denominadas **servidores**. A menudo estos servidores están alojados en una ubicación central y un administrador de sistemas se encarga de su mantenimiento. Por el contrario, los empleados tienen en sus escritorios máquinas más simples conocidas como **clientes**, con las cuales acceden a los datos remotos, por ejemplo, para incluirlos en las hojas de cálculo que desarrollan (algunas veces nos referiremos al usuario humano del equipo cliente como el “cliente”, aunque el contexto debe dejar en claro si nos referimos a la computadora o a su usuario). Las máquinas cliente y servidor se conectan mediante una red, como se muestra en la figura 1-1. Observe que mostramos la red como un óvalo simple, sin ningún detalle. Utilizaremos esta forma cuando hablemos de una red en el sentido más abstracto. Proveeremos los detalles según se requieran.

A esta disposición se le conoce como **modelo cliente-servidor**. Es un modelo ampliamente utilizado y forma la base de muchas redes. La realización más popular es la de una **aplicación web**, en la cual el servidor genera páginas web basadas en su base de datos en respuesta a las solicitudes de los clientes que pueden actualizarla. El modelo cliente-servidor es aplicable cuando el cliente y el servidor se encuentran

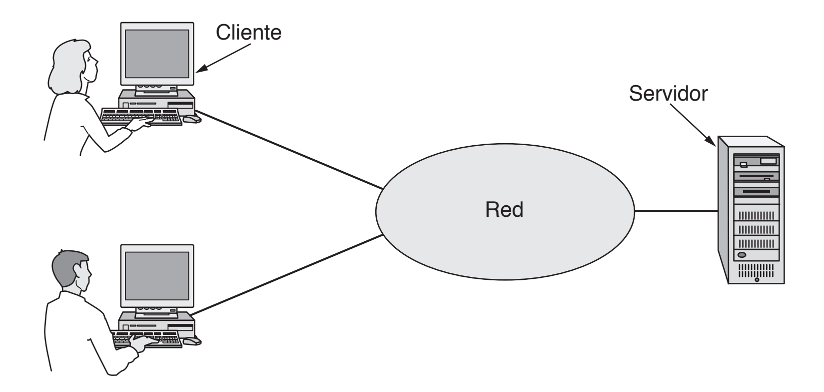


Figura 1-1. Una red con dos clientes y un servidor.

en el mismo edificio (y pertenecen a la misma empresa), pero también cuando están muy alejados. Por ejemplo, cuando una persona accede desde su hogar a una página en la World Wide Web se emplea el mismo modelo, en donde el servidor web remoto representa al servidor y la computadora personal del usuario representa al cliente. En la mayoría de las situaciones un servidor puede manejar un gran número (cientos o miles) de clientes simultáneamente.

Si analizamos detalladamente el modelo cliente-servidor, podremos ver que hay dos procesos (es decir, programas en ejecución) involucrados: uno en la máquina cliente y otro en la máquina servidor. La comunicación ocurre cuando el proceso cliente envía un mensaje a través de la red al proceso servidor. El proceso cliente espera un mensaje de respuesta. Cuando el proceso servidor obtiene la solicitud, lleva a cabo la tarea solicitada o busca los datos solicitados y devuelve una respuesta. Estos mensajes se muestran en la figura 1-2.

Un segundo objetivo al establecer una red de computadoras se relaciona con las personas y no con la información o con las computadoras. Una red de computadoras puede proveer un poderoso **medio de comunicación** entre los empleados. Ahora casi todas las empresas que tienen dos o más computadoras usan el **email** (**correo electrónico**), generalmente para la comunicación diaria. De hecho, una de las quejas comunes que se escucha por parte de los empleados a la hora de sus descansos es la gran cantidad de correos electrónicos con la que tienen que lidiar, pues la mayoría son sin sentido debido a que los jefes han descubierto que pueden enviar el mismo mensaje (a menudo sin contenido) a todos sus subordinados con sólo oprimir un botón.

En algunos casos, las llamadas telefónicas entre los empleados se pueden realizar a través de la red de computadoras en lugar de usar la compañía telefónica. A esta tecnología se le conoce como **telefonía IP** o **Voz sobre IP** (**VoIP**) cuando se utiliza la tecnología de Internet. El micrófono y el altavoz en cada extremo pueden ser de un teléfono habilitado para VoIP o la computadora del empleado. Para las empresas ésta es una maravillosa forma de ahorrar en sus cuentas telefónicas.

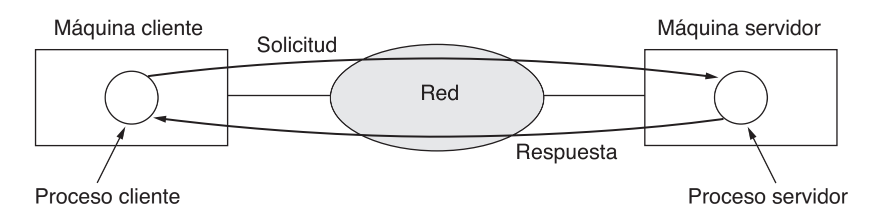


Figura 1-2. El modelo cliente-servidor implica solicitudes y respuestas.

Las redes de computadoras hacen posibles otras formas de comunicación más completas. Se puede agregar video al audio de manera que los empleados en ubicaciones distantes se puedan ver y escuchar mientras sostienen una reunión. Esta técnica es una poderosa herramienta para eliminar el costo y el tiempo dedicados a viajar. Los **escritorios compartidos** permiten a los trabajadores remotos ver una pantalla gráfica de computadora e interactuar con ella. Gracias a ello es posible que dos o más personas que trabajan a distancia lean y escriban en un pizarrón compartido, o escriban juntos un informe. Cuando un empleado realiza una modificación en un documento en línea, los demás pueden ver esa modificación de inmediato, en vez de tener que esperar varios días para recibir una carta. Dicha agilización facilita la cooperación entre los grupos remotos de personas, lo cual antes hubiera sido imposible. Hasta ahora se están empezando a utilizar formas más ambiciosas de coordinación remota como la telemedicina (por ejemplo, el monitoreo remoto de pacientes), lo cual puede tomar aún más importancia en un futuro cercano. En ocasiones se dice que la comunicación y el transporte están en constante competencia, y quien resulte ganador hará que el perdedor se vuelva obsoleto.

Un tercer objetivo para muchas empresas es realizar negocios electrónicamente, en especial con los clientes y proveedores. A este nuevo modelo se le denomina **e-commerce** (**comercio electrónico**) y ha crecido con rapidez en los años recientes. Las aerolíneas, librerías y otros vendedores han descubierto que a muchos clientes les gusta la conveniencia de comprar desde su hogar. En consecuencia, muchas empresas proveen catálogos de sus artículos y servicios en línea, e incluso reciben pedidos en línea. Los fabricantes de automóviles, aeronaves y computadoras entre otros, compran subsistemas de una variedad de proveedores y después ensamblan las piezas. Mediante el uso de redes de computadoras, los fabricantes pueden colocar los pedidos en forma electrónica según sea necesario. Esto reduce la necesidad de tener grandes inventarios y mejora la eficiencia.

1.1.2 Aplicaciones domésticas

En 1977, Ken Olsen era presidente de Digital Equipment Corporation, en ese entonces la segunda empresa distribuidora de computadoras más importante del mundo (después de IBM). Cuando se le preguntó por qué Digital no iba a incursionar a lo grande en el mercado de las computadoras personales, dijo: “No hay motivos para que una persona tenga una computadora en su hogar”. La historia demostró lo contrario y Digital desapareció. En un principio, las personas compraban computadoras para el procesamiento de palabras y para juegos. En los últimos años, probablemente la razón más importante sea acceder a Internet. En la actualidad muchos dispositivos electrónicos para el consumidor, como los decodificadores (*set-top boxes*), las consolas de juegos y los dispositivos de radio reloj, vienen con computadoras y redes integradas, especialmente redes inalámbricas; además las redes domésticas se utilizan ampliamente para actividades de entretenimiento, como escuchar, ver y crear música, fotos y videos.

El acceso a Internet ofrece a los usuarios domésticos **conectividad** a las computadoras remotas. Al igual que en las empresas, los usuarios domésticos pueden acceder a la información, comunicarse con otras personas y comprar productos y servicios mediante el comercio electrónico. Ahora el principal beneficio se obtiene al conectarse fuera del hogar. Bob Metcalfe, el inventor de Ethernet, formuló la hipótesis de que el valor de una red es proporcional al cuadrado del número de usuarios, ya que éste es aproximadamente el número de conexiones distintas que se pueden realizar (Gilder, 1993). Esta hipótesis se conoce como la “ley de Metcalfe” y nos ayuda a explicar cómo es que la enorme popularidad de Internet se debe a su tamaño.

El acceso a la información remota puede ser de varias formas. Podemos navegar en la World Wide Web para buscar información o sólo por diversión. La información disponible puede ser de varios temas, como arte, negocios, cocina, gobierno, salud, historia, ciencia, deportes, viajes y muchos más. Hay muchas maneras de divertirse como para mencionarlas aquí, además de otras que es mejor no mencionar.

Muchos periódicos se han puesto en línea y se pueden personalizar. Por ejemplo, es posible indicarle a un periódico que queremos recibir toda la información sobre políticos corruptos, grandes incendios, celebridades envueltas en escándalos y epidemias, pero nada de fútbol. También es posible hacer que se descarguen los artículos seleccionados en nuestra computadora mientras dormimos. Mientras continúe esta tendencia, cada vez más repartidores de periódicos se quedarán sin empleo, pero a los dueños de los periódicos les gusta la idea debido a que la distribución siempre ha sido el eslabón más débil en toda la cadena de producción. Claro que para que este modelo funcione tendrán primero que averiguar cómo ganar dinero en este nuevo mundo, algo que no es muy obvio dado que los usuarios de Internet esperan que todo sea gratuito.

El siguiente paso más allá de los periódicos (además de las revistas y las publicaciones científicas) es la biblioteca digital en línea. Muchas organizaciones profesionales como la ACM (*www.acm.org*) y la Sociedad de Computación del IEEE (*www.computer.org*) ya tienen todas sus publicaciones y memorias de congresos en línea. Tal vez los lectores de libros electrónicos y las bibliotecas en línea hagan obsoletos los libros impresos. Los escépticos deben tener en cuenta el efecto que tuvo la imprenta sobre el manuscrito ilustrado medieval.

Para acceder a una gran parte de esta información se utiliza el modelo cliente-servidor, aunque hay un modelo distinto y popular para acceder a la información que recibe el nombre de comunicación de **igual a igual** (*peer-to-peer*)(Parameswaran y colaboradores, 2001). En este modelo, los individuos que forman un grupo informal se pueden comunicar con otros miembros del grupo, como se muestra en la figura 1-3. En teoría, toda persona se puede comunicar con una o más personas; no hay una división fija en clientes y servidores.

Muchos sistemas de igual a igual, como BitTorrent (Cohen, 2003) no tienen una base de datos central para el contenido. En su defecto, cada usuario mantiene su propia base de datos en forma local y provee una lista de otras personas cercanas que son miembros del sistema. Así, un nuevo usuario puede ir con cualquier miembro para ver qué información tiene y obtener los nombres de otros miembros para inspeccionar si hay más contenido y más nombres. Este proceso de búsqueda se puede repetir de manera indefinida para crear una amplia base de datos local de lo que hay disponible en la red. Es una actividad que sería tediosa para las personas, pero para las computadoras es muy simple.

La comunicación de igual a igual se utiliza con frecuencia para compartir música y videos. Su mayor impacto fue en 2000 con un servicio de compartición de música llamado Napster, el cual se desmanteló después de lo que tal vez haya sido el caso de infracción de derechos de autor más grande de la historia que se haya documentado (Lam y Tan, 2001; y Macedonia, 2000). También existen aplicaciones legales

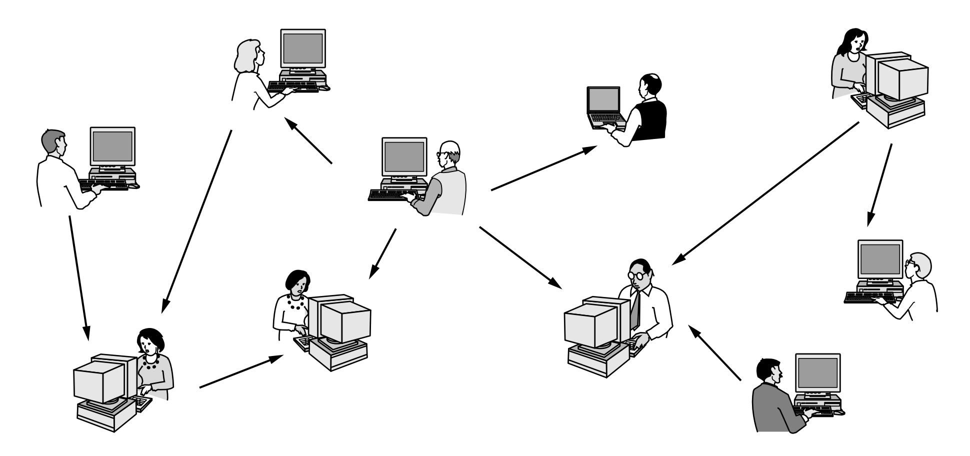


Figura 1-3. En un sistema de igual a igual no hay clientes y servidores fijos.

para la comunicación de igual a igual, como los fanáticos que comparten música del dominio público, las familias que comparten fotografías y películas caseras, y los usuarios que descargan paquetes de software públicos. De hecho, una de las aplicaciones más populares de Internet, el correo electrónico, es sin duda una aplicación de comunicación de igual a igual. Y es probable que esta forma de comunicación crezca de manera considerable en lo futuro.

Todas las aplicaciones antes mencionadas implican interacciones entre una persona y una base de datos remota llena de información. La segunda categoría importante de uso de redes es la comunicación de persona a persona, lo cual es básicamente la respuesta del siglo XXI al teléfono del siglo XIX. En la actualidad hay millones de personas en todo el mundo que utilizan el correo electrónico a diario y su uso se está extendiendo con rapidez. Es muy común que contenga audio y video, así como texto e imágenes. Tal vez la capacidad de oler los correos electrónicos todavía tarde un poco.

Todo adolescente que se precie de serlo es un adicto a la **mensajería instantánea**. Esta herramienta, que se deriva del programa *talk* de UNIX, se utiliza desde la década de 1970 y permite que dos personas se escriban mensajes entre sí en tiempo real. También hay servicios de mensajes multipersonas, como **Twitter**, que permite a las personas enviar mensajes cortos de texto, denominados *tweets*, a su círculo de amigos o cualquier otra audiencia dispuesta a recibirlos.

Las aplicaciones pueden usar Internet para transmitir audio (por ejemplo, las estaciones de radio de Internet) y video (por ejemplo, YouTube). Además de ser una forma económica de llamar a los amigos distantes, estas aplicaciones pueden proveer experiencias enriquecedoras como el teleaprendizaje, con lo cual un estudiante puede asistir a sus clases de las 8:00 a.m. sin tener que levantarse de la cama. A la larga, el uso de las redes para mejorar la comunicación de humano a humano tal vez demuestre ser más importante que cualquier otra aplicación. Quizás en el futuro sea muy importante para que las personas con inconveniencias geográficas, puedan obtener el mismo acceso a los servicios que las personas que viven en medio de una gran ciudad.

Las aplicaciones de **redes sociales** se encuentran entre las comunicaciones de persona a persona y de acceso a la información. Aquí el flujo de información se controla mediante las relaciones que las personas se declaran entre sí. Uno de los sitios de redes sociales más popular es **Facebook**. Este sitio permite a las personas actualizar sus perfiles y compartir las actualizaciones con otros que estén declarados como sus amigos. Otras aplicaciones de redes sociales pueden hacer presentaciones de amigos a través de amigos, enviar mensajes de noticias a éstos como el servicio de Twitter antes mencionado, y mucho más.

Incluso de una manera informal, grupos de personas pueden trabajar en conjunto para crear contenido. Por ejemplo, una **wiki** es un sitio web colaborativo que editan los miembros de una comunidad. La wiki más famosa es **Wikipedia**, una enciclopedia que todo el mundo puede editar, aunque hay miles de wikis más.

Nuestra tercera categoría es el comercio electrónico en el sentido más amplio del término. Las compras desde el hogar ya son populares, además de que permiten a los usuarios inspeccionar los catálogos en línea de miles de empresas. Algunos de estos catálogos son interactivos: muestran productos desde distintos puntos de vista y configuraciones que se pueden personalizar.

Si el cliente compra un producto en forma electrónica pero no puede averiguar cómo usarlo, puede obtener soporte técnico en línea.

Otra área en la cual el comercio electrónico se utiliza ampliamente es para acceder a las instituciones financieras. Muchas personas ya pagan sus facturas, administran sus cuentas bancarias y manejan sus inversiones por medios electrónicos. Es muy probable que esta tendencia continúe a medida que las redes se hagan más seguras.

Una de las áreas que casi nadie pudo prever es la de los "mercados de pulgas" electrónicos (bazares). Las subastas en línea de artículos de segunda mano se han convertido en una industria inmensa. A diferencia del comercio electrónico tradicional que sigue el modelo cliente-servidor, las subastas en línea son de igual a igual en cuanto a que los consumidores pueden actuar como compradores y como vendedores.

Algunas de estas formas de comercio electrónico han adquirido pequeñas e ingeniosas etiquetas debido al hecho de que la palabra "to" y el número "2" en inglés se pronuncian igual. En la figura 1-4 se muestra una lista de las más populares.

Etiqueta	Nombre completo	Ejemplo
B2C	Negocio a consumidor (*Business-to-consumer*)	Pedir libros en línea.
B2B	Negocio a negocio (*Business-to-business*)	Un fabricante de autos que pide los neumáticos al proveedor.
G2C	Gobierno a consumidor (*Government-to-consumer*)	El gobierno que distribuye formatos fiscales vía electrónica.
C2C	Consumidor a consumidor (*Consumer-to-consumer*)	Subastar productos de segunda mano en línea.
P2P	Igual a igual (*Peer-to-peer*)	Compartir música.

Figura 1-4. Algunas formas de comercio electrónico.

El entretenimiento es la cuarta categoría. Éste ha hecho grandes progresos en el hogar en años recientes gracias a la distribución de música, programas de radio y televisión, además de que las películas a través de Internet empiezan a competir con los mecanismos tradicionales. Los usuarios pueden buscar, comprar y descargar canciones en MP3 y películas con calidad de DVD para agregarlas a su colección personal. Los programas de TV ahora llegan a muchos hogares por medio de sistemas **IPTV** (**TeleVisión IP**) basados en la tecnología IP en vez de las transmisiones de radio o TV por cable. Las aplicaciones de flujo continuo de medios (*streaming*) permiten a los usuarios sintonizar estaciones de radio por Internet o ver los episodios recientes de sus programas favoritos de TV. Naturalmente, es posible mover todo este contenido por todo el hogar entre distintos dispositivos, pantallas y bocinas, por lo general a través de una red inalámbrica.

En un futuro cercano tal vez sea posible buscar cualquier película o programa de televisión que se haya realizado en cualquier país y hacer que se despliegue en nuestra pantalla al instante. Las nuevas películas tal vez se hagan interactivas, en donde ocasionalmente se le pida al usuario que decida el curso de la historia (¿debería Macbeth asesinar a Duncan o sólo esperar a que le llegue la hora?) y existan escenarios para todos los posibles casos. Probablemente la televisión en vivo también se vuelva interactiva, de manera que la audiencia pueda participar en los programas de preguntas, elegir de entre varios competidores, etcétera.

Los juegos son otra forma más de entretenimiento. Ya existen los juegos de simulación multipersonas en tiempo real, como jugar a las escondidas en un calabozo virtual, y los simuladores de vuelo en donde los jugadores de un equipo tratan de derribar a los jugadores del equipo contrario. Los mundos virtuales ofrecen un entorno persistente en donde miles de usuarios pueden experimentar una realidad compartida con gráficos tridimensionales.

Nuestra última categoría es la de la **computación ubicua**, en donde la computación está integrada a la vida diaria, como en la visión de Mark Weiser (1991). Muchos hogares ya cuentan con sistemas de seguridad que incluyen sensores de puertas y ventanas, y hay muchos sensores más que se pueden conectar a un monitor inteligente en el hogar, como los sensores de consumo de energía. Los medidores de electricidad, gas y agua podrían reportar su consumo a través de la red. Esto ahorraría dinero, ya que no habría necesidad de enviar personas para tomar la lectura de los medidores. Y nuestros detectores de humo podrían llamar al departamento de bomberos en vez de hacer un ruido ensordecedor (que tiene poco valor si no hay nadie en casa). A medida que disminuye el costo de los sensores y las comunicaciones, se realizarán cada vez más mediciones y reportes por medio de las redes.

Cada vez hay más dispositivos electrónicos conectados en red. Por ejemplo, algunas cámaras de gama alta ya cuentan con capacidad para conectarse a una red inalámbrica para enviar fotografías a una pantalla cercana y verlas. Los fotógrafos de deportes profesionales pueden también enviar fotos a sus editores en tiempo real, primero vía inalámbrica a un punto de acceso y después a través de Internet. Los dispositivos como las televisiones que se conectan a la toma de corriente en la pared pueden usar **redes por el cableado eléctrico** para enviar información por toda la casa a través de los cables que llevan electricidad. Tal vez no sea muy sorprendente tener estos objetos en la red, pero los objetos que no consideramos como computadoras también pueden detectar y comunicar información. Por ejemplo, su regadera podría registrar el consumo de agua, proporcionarle retroalimentación visual mientras se enjabona e informar a una aplicación de monitoreo ambiental en el hogar cuando termine para ayudarle a ahorrar en su factura de agua.

Hay una tecnología conocida como **RFID** (**Identificación por Radio-Frecuencia**, del inglés *Radio Frequency IDentification*) que llevará esta idea aún más lejos. Las etiquetas RFID son chips pasivos (es decir, no tienen batería) del tamaño de estampillas y ya es posible fijarlos a libros, pasaportes, mascotas, tarjetas de crédito y demás artículos en el hogar y fuera de él. Esto permite a los lectores RFID localizar los artículos y comunicarse con ellos a una distancia de hasta varios metros, dependiendo del tipo de RFID. En un principio la RFID se comercializó para reemplazar los códigos de barras. No ha tenido éxito aún debido a que los códigos de barras son gratuitos y las etiquetas RFID cuestan unos cuantos centavos. Desde luego que las etiquetas RFID ofrecen mucho más y su precio está bajando rápidamente. Tal vez conviertan el mundo real en la Internet de cosas (ITU, 2005).

1.1.3 Usuarios móviles

Las computadoras móviles como las laptops y las computadoras de bolsillo son uno de los segmentos de más rápido crecimiento en la industria de las computadoras. Sus ventas ya han sobrepasado a las de las computadoras de escritorio. ¿Por qué querría alguien una de ellas? Con frecuencia las personas que pasan mucho tiempo fuera de su oficina u hogar desean usar sus dispositivos móviles para leer y enviar correos electrónicos, usar Twitter, ver películas, descargar música, jugar o simplemente navegar en la Web para buscar información. Quieren hacer todas las cosas que hacen en su hogar y en su oficina. Por ende, quieren hacerlo desde cualquier lugar, ya sea en tierra, en el mar o incluso en el aire.

Muchos de estos usuarios móviles permiten la **conectividad** a Internet. Como es imposible tener una conexión alámbrica en los autos, botes y aviones, hay mucho interés en las redes móviles. Las redes celulares operadas por las compañías telefónicas son un tipo conocido de red inalámbrica que nos ofrece cobertura para los teléfonos móviles. Los **hotspots** basados en el estándar 802.11 son otro tipo de red inalámbrica para computadoras móviles. Han emergido por todos los puntos de reunión de la gente, que ahora cuenta con cobertura en cafés, hoteles, aeropuertos, trenes y aviones. Cualquiera con una laptop y un módem inalámbrico sólo necesita encender su computadora para estar conectado a Internet por medio de un hotspot, como si la computadora estuviera conectada a una red alámbrica.

Las redes inalámbricas son de gran valor para las flotillas de camiones, taxis, vehículos de reparto y técnicos para mantenerse en contacto con su base. Por ejemplo, en muchas ciudades los taxistas son comerciantes independientes, más que trabajar como empleados de una compañía de taxis. En algunas de estas ciudades los taxis tienen una pantalla que el conductor puede ver. Cuando llama un cliente, un despachador central introduce los puntos de partida y de destino. Esta información se despliega en las pantallas de los conductores y suena un timbre. El primer conductor en oprimir un botón en la pantalla obtiene la llamada.

Las redes inalámbricas también son importantes para los militares. Si de repente usted tiene que pelear una guerra en cualquier parte de la Tierra, probablemente no sea buena idea confiar en que podrá usar la infraestructura de red local. Es mejor que lleve su propia red.

Aunque es común que las redes inalámbricas y la computación móvil estén relacionadas, no son idénticas como lo muestra la figura 1-5. Aquí podemos ver una distinción entre las redes **inalámbricas fijas** y las redes **inalámbricas móviles**. Incluso las computadoras tipo notebook se conectan algunas veces mediante un cable de red. Por ejemplo, si un viajero conecta una computadora notebook al cable de red alámbrica en un cuarto de hotel, obtiene movilidad sin necesidad de una red inalámbrica.

Inalámbrica	Móvil	Aplicaciones comunes
No	No	Computadoras de escritorio en oficinas.
No	Sí	Una computadora notebook que se utiliza en un cuarto de hotel.
Sí	No	Las redes en edificios sin cables.
Sí	Sí	El inventario de la tienda con una computadora de mano.

Figura 1-5. Combinaciones de redes inalámbricas y computación móvil.

En contraste, algunas computadoras inalámbricas no son móviles. En el hogar y en las oficinas u hoteles que carecen de un cableado adecuado, puede ser más conveniente conectar las computadoras de escritorio o los reproductores de medios en forma inalámbrica en vez de instalar cables. Para instalar una red inalámbrica sólo hay que comprar una pequeña caja con ciertos componentes electrónicos en su interior, desempacarla, conectarla y quizás haya que configurar algunos detalles sencillos en los equipos de cómputo. Esta solución puede ser mucho más económica que contratar trabajadores para que coloquen ductos y cables en el edificio.

Por último, también hay aplicaciones verdaderamente móviles e inalámbricas, como cuando las personas caminan por las tiendas con computadoras de mano registrando el inventario. En muchos aeropuertos concurridos, los empleados de los negocios de renta de autos trabajan en el lote de estacionamiento con computadoras móviles inalámbricas; escanean los códigos de barras o chips RFID de los autos que regresan y su dispositivo móvil, que tiene una impresora integrada, llama a la computadora principal, obtiene la información sobre la renta e imprime la factura en ese instante.

Podemos considerar al teléfono móvil como el impulsor clave de las aplicaciones móviles e inalámbricas. La **mensajería de texto** o Servicio de Mensajes Cortos (SMC) es en extremo popular, ya que permite al usuario de un teléfono móvil escribir un mensaje corto de texto que es entregado a través de la red celular a otro suscriptor móvil. Pocas personas hubieran predicho hace 10 años la gigantesca mina de oro que representa para las compañías telefónicas el hecho de que los adolescentes escriban tediosamente mensajes cortos de texto en teléfonos móviles. Pero el servicio de mensajes cortos es muy rentable, ya que a la compañía de telefonía celular le cuesta una pequeña fracción de un centavo transmitir un mensaje de texto, servicio por el cual cobran mucho más que eso.

Por fin ha llegado la tan esperada convergencia de los teléfonos e Internet; esto acelerará el crecimiento de las aplicaciones móviles. Los **teléfonos inteligentes** (como el popular iPhone) combinan los aspectos de los teléfonos y las computadoras móviles. Las redes celulares (3G y 4G) a las cuales se conectan pueden ofrecer servicios de datos rápidos para usar Internet y manejar a la vez las llamadas telefónicas. Muchos teléfonos avanzados se conectan también a los *hotspots* inalámbricos y cambian de una red a otra en forma automática para elegir la mejor opción disponible para el usuario.

Existen otros dispositivos electrónicos que también pueden usar las redes celulares y los *hotspots* de manera que puedan permanecer conectados con computadoras remotas. Los lectores de libros electrónicos pueden descargar un libro recién comprado o la siguiente edición de una revista o del periódico de hoy, en cualquier lugar en el que se encuentren. Los portarretratos electrónicos pueden actualizar sus pantallas al instante con nuevas imágenes.

Dado que los teléfonos móviles pueden ubicarse gracias a que comúnmente están equipados con receptores **GPS** (**Sistema de Posicionamiento Global**, del inglés *Global Positioning System*), algunos de sus servicios dependen de la ubicación. Los mapas móviles y las indicaciones son el ejemplo más obvio, ya que es probable que su teléfono y automóvil habilitados con GPS tengan mejor capacidad que usted para averiguar dónde está ubicado en un momento dado. Otros ejemplos podrían ser buscar una biblioteca o un restaurante chino que esté cerca, o el pronóstico del clima local. Hay otros servicios que pueden registrar la ubicación, como al incluir en las fotos y videos una anotación del lugar donde se tomaron. A esta anotación se le conoce como "geoetiquetado".

El **comercio-m** (**comercio móvil**) es un área en la que los teléfonos móviles están comenzando a utilizarse (Senn, 2000). Los mensajes cortos de texto del dispositivo móvil se utilizan para autorizar pagos de alimentos en las máquinas expendedoras, boletos del cine y otros artículos pequeños en vez de usar efectivo y tarjetas de crédito. Posteriormente el cargo aparece en la factura del teléfono celular. Cuando el dispositivo móvil está equipado con tecnología **NFC** (**Comunicación de Campo Cercano**, del inglés *Near Field Comunication*), puede actuar como una tarjeta inteligente RFID e interactuar con un lector cercano para realizar un pago. Las fuerzas motrices detrás de este fenómeno son los fabricantes de dispositivos móviles y los operadores de red, que hacen su mejor esfuerzo por tratar de averiguar cómo obtener una rebanada del pastel del comercio electrónico. Desde el punto de vista de la tienda, este esquema les puede ahorrar la mayor parte de la cuota de las compañías de tarjetas de crédito, que puede ser del uno por ciento o mayores. Claro que este plan podría fracasar debido a que los clientes en una tienda podrían usar los lectores de código de barras o RFID en sus dispositivos móviles para verificar los precios de la competencia antes de comprar, y también podrían usarlos para obtener un informe detallado sobre la ubicación y precios de la tienda más cercana.

Una de las grandes ventajas del comercio-m es que los usuarios de teléfonos móviles están acostumbrados a pagar por todo (en contraste a los usuarios de Internet, quienes esperan que todo sea gratuito). Si un sitio web en Internet cobrara una cuota por permitir a sus clientes pagar con tarjeta de crédito, habría muchas quejas por parte de los usuarios. No obstante, si una compañía de telefonía móvil permitiera a sus clientes pagar por los artículos en una tienda con sólo ondear su teléfono frente a la caja registradora y después les cobrara una cuota por ese servicio, probablemente los usuarios lo aceptarían como algo normal. El tiempo nos lo dirá.

Sin duda, el número de usuarios de computadoras móviles e inalámbricas aumentará con rapidez en el futuro a medida que se reduzca el tamaño de éstas, probablemente en formas que nadie puede prever por ahora. Demos un vistazo a algunas posibilidades. Las **redes de sensores** están compuestas por nodos que recopilan y transmiten en forma inalámbrica la información que detectan sobre el estado del mundo físico. Los nodos pueden ser parte de elementos conocidos, como autos o teléfonos, o pueden ser pequeños dispositivos independientes. Por ejemplo, su automóvil podría recopilar la información sobre su ubicación, velocidad, vibración y ahorro de combustible desde su sistema de diagnóstico integrado y enviar esta información a una base de datos (Hull y colaboradores, 2006). Esos datos pueden ayudar a encontrar baches, planear viajes alrededor de caminos congestionados e indicarnos si somos unos "devoradores de gasolina" en comparación con otros conductores en la misma extensión del camino.

Las redes de sensores están revolucionando la ciencia al proveer una gran cantidad de datos sobre el comportamiento, lo cual no era posible observar antes. Como ejemplo podemos mencionar el rastreo individual de cebras durante su migración, al colocar un pequeño sensor en cada animal (Juang y colaboradores, 2002). Los investigadores han logrado empacar una computadora inalámbrica en un cubo de 1 mm de grosor (Warneke y colaboradores, 2001). Con computadoras móviles así de pequeñas podemos rastrear incluso hasta aves pequeñas, roedores e insectos.

A lo anterior le podemos dar incluso usos triviales (como en los parquímetros), ya que se utilizan datos que no estaban disponibles antes. Los parquímetros inalámbricos pueden aceptar pagos con tarjetas de crédito o débito con verificación instantánea a través del enlace inalámbrico. También pueden reportar

cuando estén en uso mediante la red inalámbrica. Esto permitiría a los conductores descargar un mapa de parquímetros reciente en su auto, para que puedan encontrar un lugar disponible con más facilidad. Claro que al expirar, un parquímetro podría también verificar la presencia de un automóvil (al enviar una señal y esperar su rebote) y reportar a las autoridades de tránsito su expiración. Se estima que en Estados Unidos tan sólo los gobiernos municipales podrían recolectar unos $10 mil millones de dólares adicionales de esta forma (Harte y colaboradores, 2000).

Las **computadoras usables** son otra aplicación prometedora. Los relojes inteligentes con radio han formado parte de nuestro espacio mental desde que aparecieron en la tira cómica de Dick Tracy, en 1946, ahora es posible comprarlos. También hay otros dispositivos de este tipo que se pueden implementar, como los marcapasos y las bombas de insulina. Algunos de ellos se pueden controlar a través de una red inalámbrica. Esto permitiría a los doctores probarlos y reconfigurarlos con más facilidad. Incluso podrían surgir graves problemas si los dispositivos fueran tan inseguros como la PC promedio y alguien pudiera intervenirlos fácilmente (Halperin y colaboradores, 2008).

1.1.4 Cuestiones sociales

Al igual que la imprenta hace 500 años, las redes de computadoras permiten a los ciudadanos comunes distribuir y ver el contenido en formas que no hubiera sido posible lograr antes. Pero con lo bueno viene lo malo, y esta posibilidad trae consigo muchas cuestiones sociales, políticas y éticas sin resolver; a continuación mencionaremos brevemente algunas de ellas, ya que para un estudio completo de las mismas se requeriría por lo menos todo un libro.

Las redes sociales, los tableros de mensajes, los sitios de compartición de contenido y varias aplicaciones más permiten a las personas compartir sus opiniones con individuos de pensamientos similares. Mientras que los temas estén restringidos a cuestiones técnicas o aficiones como la jardinería, no surgirán muchas dificultades.

El verdadero problema está en los temas que realmente importan a las personas, como la política, la religión y el sexo. Hay opiniones que si se publican y quedan a la vista de todos pueden ser bastante ofensivas para algunas personas. O peor aún, tal vez no sean políticamente correctas. Lo que es más, las opiniones no necesitan limitarse sólo a texto; es posible compartir fotografías a color de alta resolución y clips de video a través de las redes de computadoras. Algunas personas toman una posición del tipo “vive y deja vivir”, pero otras sienten que simplemente es inaceptable publicar cierto material (como ataques verbales a países o religiones específicas, pornografía, etc.) y que es necesario censurar dicho contenido. Cada país tiene diferentes leyes contradictorias sobre este tema. Por ende, el debate se aviva.

En el pasado reciente las personas demandaban a los operadores de red afirmando que eran responsables por el contenido de lo que transmitían, al igual que los periódicos y las revistas. La respuesta inevitable es que una red es como una compañía telefónica o la oficina postal, por lo que no es posible que esté vigilando lo que sus usuarios dicen.

Para estos momentos tal vez le sorprenda un poco saber que algunos operadores de red bloquean contenido por motivos personales. Algunos suspendieron el servicio de red a varios usuarios de aplicaciones de igual a igual debido a que no consideraron rentable transmitir las grandes cantidades de tráfico que envían esas aplicaciones. Probablemente estos mismos operadores traten a las diversas empresas de manera diferente. Si usted es una empresa grande y paga bien, recibe un buen servicio, pero si es un comerciante pequeño recibirá un mal servicio. Los que se oponen a esta práctica argumentan que el contenido de las redes de igual a igual y cualquier otro tipo de contenido debe tratarse de la misma forma, ya que son sólo bits en la red. A este argumento que sostiene que no hay que diferenciar las comunicaciones según su contenido u origen, o con base en quién lo provee, se le conoce como **neutralidad de red** (Wu, 2003). Es muy probable que este debate persista por mucho tiempo.

Hay muchas otras partes involucradas en la lucha sobre el contenido. Por ejemplo, la música y las películas piratas impulsaron el crecimiento masivo de las redes de igual a igual, lo cual no agradó a los dueños de los derechos de autor, quienes han amenazado con tomar (y algunas veces han tomado) acción legal. Ahora hay sistemas automatizados que buscan redes de igual a igual y envían advertencias a los operadores de red y usuarios sospechosos de infringir los derechos de autor. En Estados Unidos a estas advertencias se les conoce como **avisos de DCMA para quitar contenido** según la **Ley de Copyright del Milenio Digital**. Esta búsqueda es una carrera armamentista, ya que es difícil detectar de manera confiable el momento en que se violan los derechos de autor. Incluso hasta su impresora podría ser considerada como culpable (Piatek y colaboradores, 2008).

Las redes de computadoras facilitan considerablemente la comunicación. También ayudan a las personas que operan la red con el proceso de husmear en el tráfico. Esto provoca conflictos sobre cuestiones como los derechos de los empleados frente a los derechos de los patrones. Muchas personas leen y escriben correos electrónicos en su trabajo. Muchos patrones han reclamado el derecho de leer y tal vez censurar los mensajes de los empleados, incluyendo los mensajes enviados desde una computadora en el hogar, después de las horas de trabajo. No todos los empleados están de acuerdo con esto, en especial con lo último.

Otro conflicto se centra alrededor de los derechos del gobierno frente a los derechos de los ciudadanos. El FBI ha instalado sistemas con muchos proveedores de servicios de Internet para analizar todo el correo electrónico entrante y saliente en busca de fragmentos que le interesen. Uno de los primeros sistemas se llamaba originalmente Carnivore, pero la mala publicidad provocó que cambiaran su nombre por el de DCS1000, algo más inocente (Blaze y Bellovin, 2000; Sobel, 2001 y Zacks, 2001). El objetivo de este sistema es espiar a millones de personas con la esperanza de encontrar información sobre actividades ilegales. Por desgracia para los espías, la Cuarta Enmienda a la Constitución de Estados Unidos prohíbe las búsquedas gubernamentales sin una orden de cateo, pero a menudo el gobierno ignora esta regulación.

Claro que el gobierno no es el único que amenaza la privacidad de las personas. El sector privado también participa al crear **perfiles** de los usuarios. Por ejemplo, los pequeños archivos llamados **cookies** que los navegadores web almacenan en las computadoras de los usuarios permiten a las empresas rastrear las actividades de los usuarios en el ciberespacio y también pueden permitir que los números de tarjetas de crédito, de seguro social y demás información confidencial se filtren por todo Internet (Berghel, 2001). Las empresas que proveen servicios basados en web pueden mantener grandes cantidades de información personal sobre sus usuarios para estudiar directamente sus actividades. Por ejemplo, Google puede leer su correo electrónico y mostrarle anuncios basados en sus intereses si utiliza su servicio de correo electrónico **Gmail**.

Un nuevo giro en el ámbito de los dispositivos móviles es la privacidad de la ubicación (Beresford y Stajano, 2003). Como parte del proceso de proveer servicio a los dispositivos móviles, los operadores de red aprenden en dónde se encuentran los usuarios a distintas horas del día. Esto les permite rastrear sus movimientos. Tal vez sepan qué club nocturno frecuenta usted y a cuál centro médico asiste.

Las redes de computadoras también ofrecen el potencial de incrementar la privacidad al enviar mensajes anónimos. En ciertas situaciones, esta capacidad puede ser conveniente. Además de evitar que las empresas conozcan los hábitos de sus clientes, también ofrece, por ejemplo, los medios para que los estudiantes, soldados, empleados y ciudadanos puedan denunciar el comportamiento ilegal por parte de profesores, oficiales, superiores y políticos sin temor a las represalias. Por otra parte, en Estados Unidos, y en la mayoría de otras democracias, la ley permite de manera específica que una persona acusada tenga el derecho de confrontar y desafiar a su acusador en la corte, por lo que no se permite el uso de acusaciones anónimas como evidencia.

Internet hace posible encontrar información rápidamente, pero gran parte de ella se considera de dudosa procedencia, engañosa o en definitiva incorrecta. Ese consejo médico que usted obtuvo de Internet en relación con el dolor en su pecho puede haber provenido de un ganador del Premio Nobel o de un chico sin estudios.

Hay otro tipo de información que por lo general es indeseable. El correo electrónico basura (*spam*) se ha convertido en parte de la vida, ya que los emisores de correo electrónico basura (*spammers*) han recolectado millones de direcciones de correo electrónico y los aspirantes a vendedores pueden enviarles mensajes generados por computadora a un costo muy bajo. La inundación resultante de spam rivaliza con el flujo de mensajes de personas reales. Por fortuna hay software de filtrado capaz de leer y desechar el *spam* generado por otras computadoras, aunque su grado de éxito puede variar en forma considerable.

Existe también contenido destinado al comportamiento criminal. Las páginas web y los mensajes de correo electrónico con contenido activo (en esencia, programas o macros que se ejecutan en la máquina del receptor) pueden contener virus que invadan nuestra computadora. Podrían utilizarlos para robar las contraseñas de nuestras cuentas bancarias o hacer que nuestra computadora envíe *spam* como parte de una red zombie (**botnet**) o grupo de equipos comprometidos.

Los mensajes de **suplantación de identidad** o estafas se enmascaran como si se originaran desde un sitio de confianza (como su banco, por ejemplo) para ver si el receptor les revela información delicada, como los números de sus tarjetas de crédito. El robo de identidad se está convirtiendo en un problema grave, a medida que los ladrones recolectan suficiente información sobre una víctima para obtener tarjetas de crédito y otros documentos a su nombre.

Puede ser difícil evitar que las computadoras se hagan pasar por personas en Internet. Este problema ha originado el desarrollo de **cuadros de captura de texto para verificación** (**CAPTCHAs**), en donde una computadora pide a una persona que resuelva una pequeña tarea de reconocimiento; por ejemplo, escribir las letras que se muestran en una imagen distorsionada para demostrar que son humanos (von Ahn, 2001). Este proceso es una variación de la famosa prueba de Turing, en donde una persona hace preguntas a través de una red para juzgar si la entidad que responde es humana.

Podríamos resolver muchos de estos problemas si la industria de la computación tomara en serio la seguridad de las computadoras. Si se cifraran y autenticaran todos los mensajes, sería más difícil tener dificultades. Dicha tecnología está bien establecida y la estudiaremos con detalle en el capítulo 8. El inconveniente es que los distribuidores de hardware y software saben que es costoso incluir herramientas de seguridad y sus clientes no exigen dichas características. Además, una gran cantidad de los problemas son provocados por el software defectuoso, ya que los distribuidores siguen agregando cada vez más características a sus programas, lo cual se traduce inevitablemente en más código y por ende más errores. Tal vez sería conveniente aplicar un impuesto para las nuevas características, pero no todos estarían convencidos de que sea la mejor solución. También sería agradable que hubiera un reembolso por el software defectuoso pero, de ser así, toda la industria del software quedaría en bancarrota en menos de un año.

Las redes de computadoras generan nuevos problemas legales cuando interactúan con las antiguas leyes. Las apuestas electrónicas son un ejemplo de ello. Si las computadoras han estado simulando cosas por décadas, ¿por qué no simular máquinas tragamonedas, ruletas, repartidores de blackjack y demás equipo para apostar? Bueno, porque es ilegal en muchos lugares. El problema es que las apuestas son legales en otras partes (en Inglaterra, por ejemplo) y los propietarios de casinos de esos lugares han captado el potencial de las apuestas por Internet. Pero, ¿qué ocurriría si el apostador, el casino y el servidor estuvieran todos en distintos países, con leyes contradictorias? Buena pregunta.

1.2 HARDWARE DE RED

Ahora es tiempo de dejar a un lado las aplicaciones y los aspectos sociales de las redes para enfocarnos en las cuestiones técnicas implicadas en su diseño. No existe una clasificación aceptada en la que encajen todas las redes, pero hay dos que sobresalen de manera importante: la tecnología de transmisión y la escala. Examinaremos ahora cada una de ellas por turno.

Hablando en sentido general, existen dos tipos de tecnología de transmisión que se emplean mucho en la actualidad: los enlaces de **difusión** (*broadcast*) y los enlaces de **punto a punto**.

Los enlaces de punto a punto conectan pares individuales de máquinas. Para ir del origen al destino en una red formada por enlaces de punto a punto, los mensajes cortos (conocidos como **paquetes** en ciertos contextos) tal vez tengan primero que visitar una o más máquinas intermedias. A menudo es posible usar varias rutas de distintas longitudes, por lo que es importante encontrar las más adecuadas en las redes de punto a punto. A la transmisión punto a punto en donde sólo hay un emisor y un receptor se le conoce como **unidifusión** (*unicasting*).

Por el contrario, en una red de difusión todas las máquinas en la red comparten el canal de comunicación; los paquetes que envía una máquina son recibidos por todas las demás. Un campo de dirección dentro de cada paquete especifica a quién se dirige. Cuando una máquina recibe un paquete, verifica el campo de dirección. Si el paquete está destinado a la máquina receptora, ésta procesa el paquete; si el paquete está destinado para otra máquina, sólo lo ignora.

Una red inalámbrica es un ejemplo común de un enlace de difusión, en donde la comunicación se comparte a través de una región de cobertura que depende del canal inalámbrico y de la máquina que va a transmitir. Como analogía considere alguien parado en una sala de juntas gritando: “Watson, ven aquí. Te necesito”. Aunque muchas personas hayan recibido (escuchado) el paquete, sólo Watson responderá; los otros simplemente lo ignorarán.

Por lo general, los sistemas de difusión también brindan la posibilidad de enviar un paquete a *todos* los destinos mediante el uso de un código especial en el campo de dirección. Cuando se transmite un paquete con este código, todas las máquinas en la red lo reciben y procesan. A este modo de operación se le conoce como **difusión** (*broadcasting*). Algunos sistemas de difusión también soportan la transmisión a un subconjunto de máquinas, lo cual se conoce como **multidifusión** (*multicasting*).

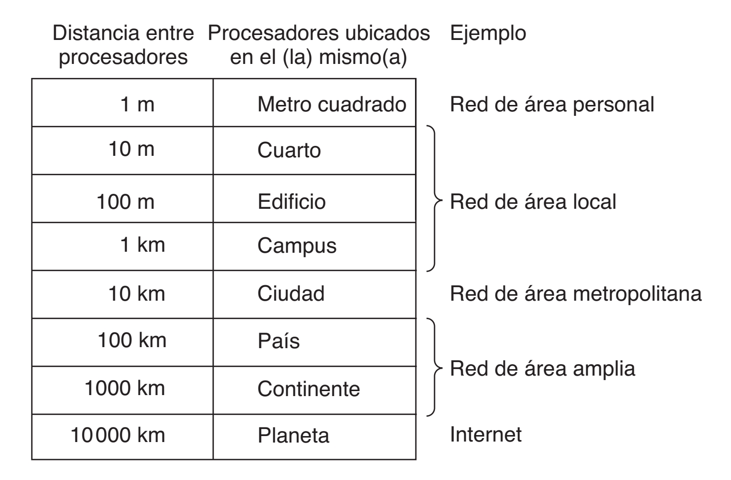
Hay un criterio alternativo para clasificar las redes: por su escala. La distancia es importante como medida de clasificación, ya que las distintas tecnologías se utilizan a diferentes escalas.

En la figura 1-6 clasificamos los sistemas multiprocesadores con base en su tamaño físico. En la parte de arriba están las redes de área personal, las cuales están destinadas a una persona. Después se encuentran redes más grandes. Éstas se pueden dividir en redes de área local, de área metropolitana y de área amplia, cada una con una escala mayor que la anterior. Por último, a la conexión de dos o más redes se le conoce como **interred** (*internetwork*). La Internet de nivel mundial es sin duda el mejor ejemplo (aunque no el único) de una interred. Pronto tendremos interredes aún más grandes con la **Internet interplanetaria** que conecta redes a través del espacio (Burleigh y colaboradores, 2003).

En este libro hablaremos sobre las redes de todas estas escalas. En las siguientes secciones le proporcionaremos una breve introducción al hardware de red con base en la escala.

1.2.1 Redes de área personal

Las **redes de área personal**, generalmente llamadas **PAN** (*Personal Area Network*) permiten a los dispositivos comunicarse dentro del rango de una persona. Un ejemplo común es una red inalámbrica que conecta a una computadora con sus periféricos. Casi todas las computadoras tienen conectado un monitor, un teclado, un ratón y una impresora. Sin la tecnología inalámbrica es necesario realizar esta conexión



Distancia entre procesadores	Procesadores ubicados en el (la) mismo(a)	Ejemplo
1 m	Metro cuadrado	Red de área personal
10 m	Cuarto	Red de área local
100 m	Edificio	
1 km	Campus	
10 km	Ciudad	Red de área metropolitana
100 km	País	Red de área amplia
1000 km	Continente	
10000 km	Planeta	Internet

Figura 1-6. Clasificación de los procesadores interconectados con base en la escala.

mediante cables. Hay tantos usuarios nuevos que batallan mucho para encontrar los cables adecuados y conectarlos en los orificios apropiados (aun cuando, por lo general, están codificados por colores), que la mayoría de los distribuidores de computadoras ofrecen la opción de enviar un técnico al hogar del usuario para que se encargue de ello. Para ayudar a estos usuarios, algunas empresas se pusieron de acuerdo para diseñar una red inalámbrica de corto alcance conocida como **Bluetooth** para conectar estos componentes sin necesidad de cables. La idea es que si sus dispositivos tienen Bluetooth, no necesitará cables. Sólo hay que ponerlos en el lugar apropiado, encenderlos y trabajarán en conjunto. Para muchas personas, esta facilidad de operación es una gran ventaja.

En su forma más simple, las redes Bluetooth utilizan el paradigma maestro-esclavo de la figura 1-7. La unidad del sistema (la PC), por lo general es el maestro que trata con el ratón, el teclado, etc., como sus esclavos. El maestro dice a los esclavos qué direcciones usar, cuándo pueden transmitir información, durante cuánto tiempo pueden transmitir, qué frecuencias usar, etcétera.

También podemos usar Bluetooth en otras aplicaciones. A menudo se utiliza para conectar unos audífonos a un teléfono móvil sin cables, además se puede conectar el reproductor musical digital a nuestro automóvil con sólo tenerlo dentro del rango. Una clase completamente distinta de red PAN se forma

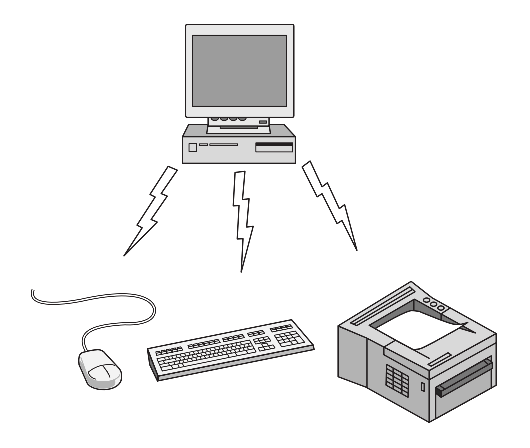


Figura 1-7. Configuración de red PAN con Bluetooth.

cuando un dispositivo médico integrado, como un marcapasos, bomba de insulina o audífono para discapacitados se comunica con un control remoto operado por el usuario. En el capítulo 4 veremos con detalle la tecnología Bluetooth.

Las redes PAN también se pueden construir con otras tecnologías que se comunican dentro de rangos cortos, como RFID en las tarjetas inteligentes y los libros de las bibliotecas. En el capítulo 4 estudiaremos la tecnología RFID.

1.2.2 Redes de área local

Las **redes de área local**, generalmente llamadas **LAN** (*Local Area Networks*), son redes de propiedad privada que operan dentro de un solo edificio, como una casa, oficina o fábrica. Las redes LAN se utilizan ampliamente para conectar computadoras personales y electrodomésticos con el fin de compartir recursos (por ejemplo, impresoras) e intercambiar información. Cuando las empresas utilizan redes LAN se les conoce como **redes empresariales**.

Las redes LAN son muy populares en la actualidad, en especial en los hogares, los edificios de oficinas antiguos, las cafeterías y demás sitios en donde es muy problemático instalar cables. En estos sistemas, cada computadora tiene un módem y una antena que utiliza para comunicarse con otras computadoras. En la mayoría de los casos, cada computadora se comunica con un dispositivo en el techo, como se muestra en la figura 1-8(a). A este dispositivo se le denomina **AP** (**Punto de Acceso**, del inglés *Access Point*), **enrutador inalámbrico** o **estación base**; transmite paquetes entre las computadoras inalámbricas y también entre éstas e Internet. El AP es como el niño popular de la escuela, ya que todos quieren hablar con él. Pero si hay otras computadoras que estén lo bastante cerca una de otra, se pueden comunicar directamente entre sí en una configuración de igual a igual.

Hay un estándar para las redes LAN inalámbricas llamado **IEEE 802.11**, mejor conocido como **WiFi**. Opera a velocidades desde 11 hasta cientos de Mbps (en este libro nos apegaremos a la tradición y mediremos las velocidades de las líneas de transmisión en megabits/segundo, en donde 1 Mbps es 1 000 000 bits/segundo, y en gigabits/segundo, en donde 1 Gbps es 1 000 000 000 bits/segundo). En el capítulo 4 hablaremos sobre el estándar 802.11.

Las redes LAN alámbricas utilizan distintas tecnologías de transmisión. La mayoría utilizan cables de cobre, pero algunas usan fibra óptica. Las redes LAN tienen restricciones en cuanto a su tamaño, lo cual significa que el tiempo de transmisión en el peor de los casos es limitado y se sabe de antemano. Conocer estos límites facilita la tarea del diseño de los protocolos de red. Por lo general las redes LAN alámbricas que operan a velocidades que van de los 100 Mbps hasta un 1 Gbps, tienen retardo bajo (microsegundos

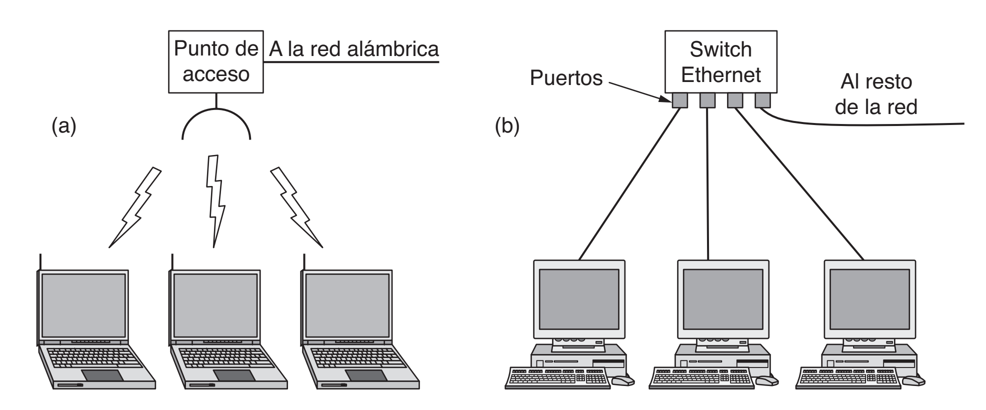


Figura 1-8. Redes inalámbrica y alámbrica. (a) 802.11. (b) Ethernet conmutada.

o nanosegundos) y cometen muy pocos errores. Las redes LAN más recientes pueden operar a una velocidad de hasta 10 Gbps. En comparación con las redes inalámbricas, las redes LAN alámbricas son mucho mejores en cuanto al rendimiento, ya que es más fácil enviar señales a través de un cable o fibra que por el aire.

La topología de muchas redes LAN alámbricas está basada en los enlaces de punto a punto. El estándar IEEE 802.3, comúnmente conocido como **Ethernet**, es hasta ahora el tipo más común de LAN alámbrica. La figura 1-8(b) muestra un ejemplo de topología de **Ethernet conmutada**. Cada computadora se comunica mediante el protocolo Ethernet y se conecta a una caja conocida como **switch** con un enlace de punto a punto. De aquí que tenga ese nombre. Un switch tiene varios **puertos**, cada uno de los cuales se puede conectar a una computadora. El trabajo del switch es transmitir paquetes entre las computadoras conectadas a él, y utiliza la dirección en cada paquete para determinar a qué computadora se lo debe enviar.

Para crear redes LAN más grandes se pueden conectar switches entre sí mediante sus puertos. ¿Qué ocurre si los conectamos en un circuito cerrado? ¿Podrá funcionar la red así? Por fortuna, los diseñadores consideraron este caso. Es responsabilidad del protocolo determinar qué rutas deben recorrer los paquetes para llegar de manera segura a la computadora de destino. En el capítulo 4 veremos cómo funciona esto.

También es posible dividir una gran LAN física en dos redes LAN lógicas más pequeñas. Tal vez se pregunte por qué sería esto útil. En ocasiones la distribución del equipo de red no coincide con la estructura de la organización. Por ejemplo, los departamentos de ingeniería y finanzas de una empresa podrían tener computadoras en la misma LAN física debido a que se encuentran en la misma ala del edificio, pero podría ser más sencillo administrar el sistema si cada departamento tuviera su propia red lógica, denominada **LAN virtual** o **VLAN**. En este diseño cada puerto se identifica con un “color”; por ejemplo, verde para ingeniería y rojo para finanzas. Después el switch reenvía los paquetes de manera que las computadoras conectadas a los puertos verdes estén separadas de las que están conectadas a los puertos rojos. Por ejemplo, los paquetes de difusión que se envíen por un puerto rojo no se recibirán en un puerto verde, tal como si hubiera dos redes LAN distintas. Al final del capítulo 4 veremos los detalles sobre las redes VLAN.

También existen otras topologías de LAN alámbrica. De hecho, la Ethernet conmutada es una versión moderna del diseño original de Ethernet en el que se difundían todos los paquetes a través de un solo cable lineal. Sólo una máquina podía transmitir con éxito en un instante dado, y se utilizaba un mecanismo de arbitraje distribuido para resolver los conflictos. Utilizaba un algoritmo simple: las computadoras podían transmitir siempre que el cable estuviera inactivo. Si ocurría una colisión entre dos o más paquetes, cada computadora esperaba un tiempo aleatorio y volvía a intentar. Llamaremos a esa versión **Ethernet clásica** por cuestión de claridad y, como tal vez se lo imagine, aprenderá sobre ella en el capítulo 4.

Las redes inalámbricas y las alámbricas se pueden dividir en diseños estáticos y dinámicos, dependiendo de la forma en que se asigna el canal. Una asignación estática típica sería dividir el tiempo en intervalos discretos y utilizar un algoritmo por turno rotatorio (*round-robin*), para que cada máquina pueda difundir los datos sólo cuando sea su turno de usar su intervalo. La asignación estática desperdicia la capacidad del canal cuando una máquina no tiene nada que decir durante su intervalo asignado, por lo que la mayoría de los sistemas tratan de asignar el canal en forma dinámica (es decir, bajo demanda).

Los métodos de asignación dinámica para un canal común pueden ser centralizados o descentralizados. En el método de asignación de canal centralizado hay una sola entidad (por ejemplo, la estación base en las redes celulares) que determina el turno de cada quien. Para ello podría aceptar varios paquetes y asignarles prioridades de acuerdo con algún algoritmo interno. En el método de asignación de canal descentralizado no hay una entidad central; cada máquina debe decidir por su cuenta si va a transmitir o no. Tal vez usted piense que esta metodología provoca un caos, pero no es así. Más adelante estudiaremos muchos algoritmos diseñados para poner orden a un potencial caos.

Vale la pena invertir un poco más de tiempo para hablar sobre las redes LAN en el hogar. En lo futuro es probable que todos los dispositivos en el hogar sean capaces de comunicarse con cualquier otro dispositivo, y todos ellos serán accesibles a través de Internet. Tal vez este acontecimiento sea uno de esos conceptos visionarios que nadie solicitó (como los controles remotos de TV o los teléfonos móviles), pero una vez que llegaron nadie se imagina cómo pudo haber vivido sin ellos.

Muchos dispositivos ya son capaces de conectarse en red. Entre ellos tenemos a las computadoras, los dispositivos de entretenimiento como las TV y los DVD, teléfonos y otros dispositivos electrónicos como las cámaras, aparatos como los radios relojes e infraestructura como los medidores de servicios y termostatos. Esta tendencia seguirá avanzando. Por ejemplo, es probable que el hogar promedio tenga una docena de relojes (es decir, en aparatos), los cuales, si estuvieran conectados a Internet, podrían ajustarse de manera automática al horario de verano para ahorrar energía solar. Es muy probable que el monitoreo remoto del hogar sea una aplicación muy popular en el futuro, ya que muchos hijos en edad adulta estarían dispuestos a invertir algo de dinero para ayudar a sus padres envejecidos a vivir con seguridad en sus propios hogares.

Aunque podríamos considerar a la red doméstica como cualquier otra LAN, es muy probable que tenga distintas propiedades. En primer lugar, los dispositivos en red tienen que ser muy fáciles de instalar. Los enrutadores inalámbricos son uno de los artículos que más devuelven los consumidores. Las personas compran uno porque desean una red inalámbrica en su hogar, pero al sacarlo de su caja descubren que no está "listo para usarse"; por lo tanto, prefieren devolverlo en lugar de esperar a ser atendidas en la línea telefónica de asistencia.

En segundo lugar, la red y los dispositivos tienen que operar en un modo a prueba de errores. Los aires acondicionados solían tener una perilla con cuatro posiciones: Apagado, bajo, medio y alto. Ahora tienen manuales de 30 páginas. Una vez que puedan conectarse en red, es probable que tan sólo el capítulo sobre seguridad sea de ese tamaño. Éste es un problema debido a que sólo los usuarios de computadoras están acostumbrados a lidiar con productos que no funcionan; el público que compra autos, televisiones y refrigeradores es menos tolerante. Esperan productos que funcionen al 100% sin tener que contratar a un experto en computadoras.

En tercer lugar, el precio es imprescindible para el éxito. Las personas no pagarán una tarifa de $50 dólares por un termostato con conexión a Internet debido a que pocas personas consideran que sea tan importante monitorear la temperatura de su hogar desde el trabajo. Aunque tal vez por $5 dólares adicionales sí podría llegar a venderse.

En cuarto lugar, debe existir la posibilidad de empezar con uno o dos dispositivos para después expandir el alcance de la red en forma gradual. Esto significa que no debe haber guerras de formatos. Decir a los consumidores que compren periféricos con interfaces IEEE 1394 (*FireWire*) para luego retractarse unos cuantos años después y decir que USB 2.0 es la interfaz del mes, y luego cambiarla por la interfaz 802.11g (¡ups!, no, mejor que sea 802.11n), o quizá mejor 802.16 (distintas redes inalámbricas), son acciones que volverán a los consumidores muy escépticos. La interfaz de red tendrá que permanecer estable por décadas, así como los estándares de transmisión por televisión.

En quinto lugar, la seguridad y la confiabilidad serán de extrema importancia. Perder unos cuantos archivos debido a un virus de correo electrónico es una cosa; que un ladrón desarme nuestro sistema de seguridad desde su computadora móvil y después saquee nuestro hogar es muy distinto.

Una pregunta interesante es si las redes domésticas serán alámbricas o inalámbricas. La conveniencia y el costo favorecen a las redes inalámbricas, ya que no hay cables qué instalar (o peor aún, reinstalar). La seguridad favorece a las redes alámbricas, ya que las ondas de radio que utilizan las redes inalámbricas pueden traspasar las paredes con facilidad. No todos se alegran al saber que los vecinos se están colgando de su conexión a Internet y leyendo su correo electrónico. En el capítulo 8 estudiaremos cómo se puede utilizar el cifrado para proveer seguridad, aunque es más fácil decirlo que hacerlo cuando los usuarios son inexpertos.

Una tercera opción que podría ser interesante es la de reutilizar las redes que ya se encuentren en el hogar. El candidato más obvio es la red formada por los cables eléctricos instalados por toda la casa. Las **redes por el cableado eléctrico** permiten difundir información por toda la casa a los dispositivos que se conectan a los tomacorrientes. De todas formas usted tiene que conectar la TV, y de esta forma puede obtener conectividad a Internet al mismo tiempo. La dificultad está en cómo llevar tanto electricidad como señales de datos al mismo tiempo. Parte de la respuesta es que estas señales utilizan distintas bandas de frecuencia.

En resumen, las redes LAN domésticas ofrecen muchas oportunidades y retos. La mayoría de estos retos se relacionan con la necesidad de que las redes sean fáciles de manejar, confiables y seguras (en especial en manos de los usuarios inexpertos), así como de bajo costo.

1.2.3 Redes de área metropolitana

Una **Red de Área Metropolitana**, o **MAN** (*Metropolitan Area Network*), cubre toda una ciudad. El ejemplo más popular de una MAN es el de las redes de televisión por cable disponibles en muchas ciudades. Estos sistemas surgieron a partir de los primeros sistemas de antenas comunitarias que se utilizaban en áreas donde la recepción de televisión por aire era mala. En esos primeros sistemas se colocaba una gran antena encima de una colina cercana y después se canalizaba una señal a las casas de los suscriptores.

Al principio estos sistemas se diseñaban con fines específicos en forma local. Después, las empresas empezaron a entrar al negocio y consiguieron contratos de los gobiernos locales para cablear ciudades completas. El siguiente paso fue la programación de televisión e incluso canales completos diseñados sólo para cable. A menudo estos canales eran altamente especializados, como canales de sólo noticias, sólo deportes, sólo cocina, sólo jardinería, etc. Pero desde su comienzo hasta finales de la década de 1990, estaban diseñados sólo para la recepción de televisión.

Cuando Internet empezó a atraer una audiencia masiva, los operadores de red de TV por cable empezaron a darse cuenta de que con unos cambios en el sistema, podían proveer servicio de Internet de dos vías en partes no usadas del espectro. En ese momento, el sistema de TV por cable empezó a transformarse, de ser una simple forma de distribuir televisión, para convertirse en una red de área metropolitana. A simple vista, una MAN podría tener la apariencia del sistema que se muestra en la figura 1-9. En esta figura podemos ver que se alimentan señales de televisión y de Internet en un amplificador de cabecera para después distribuirlas a los hogares de las personas. Volveremos a ver este tema con detalle en el capítulo 2.

Cabe mencionar que la televisión por cable no es la única MAN. Los recientes desarrollos en el acceso inalámbrico a Internet de alta velocidad han originado otra, la cual se estandarizó como IEEE 802.16 y se conoce comúnmente como **WiMAX**. Hablaremos sobre ella en el capítulo 4.

1.2.4 Redes de área amplia

Una **Red de Área Amplia**, o **WAN** (*Wide Area Network*), abarca una extensa área geográfica, por lo general un país o continente. Empezaremos nuestra discusión con las redes WAN alámbricas y usaremos el ejemplo de una empresa con sucursales en distintas ciudades.

La WAN en la figura 1-10 es una red que conecta las oficinas en Perth, Melbourne y Brisbane. Cada una de estas oficinas contiene computadoras destinadas a ejecutar programas de usuario (aplicaciones). Seguiremos el uso tradicional y llamaremos a estas máquinas **hosts**. Al resto de la red que conecta estos hosts se le denomina **subred de comunicación**, o para abreviar sólo **subred**. La tarea de la subred es transportar los mensajes de host a host, al igual que el sistema telefónico transporta las palabras (en realidad sólo los sonidos) de la persona que habla a la persona que escucha.

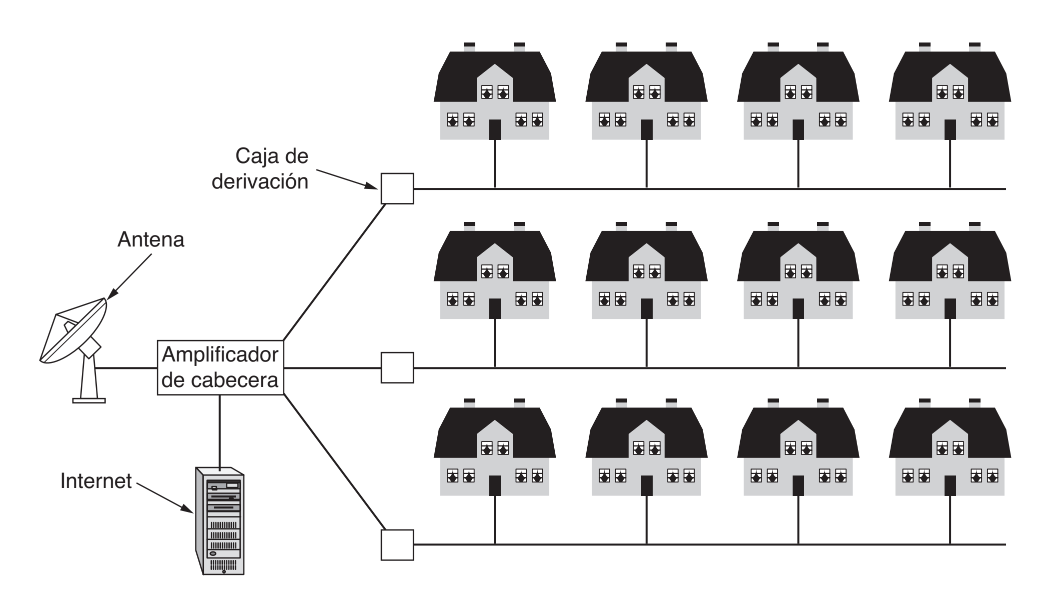


Figura 1-9. Una red de área metropolitana basada en la TV por cable.

En la mayoría de las redes WAN, la subred cuenta con dos componentes distintos: líneas de transmisión y elementos de conmutación. Las **líneas de transmisión** mueven bits entre máquinas. Se pueden fabricar a partir de alambre de cobre, fibra óptica o incluso enlaces de radio. Como la mayoría de las empresas no poseen líneas de transmisión, tienen que rentarlas a una compañía de telecomunicaciones. Los **elementos de conmutación** o **switches** son computadoras especializadas que conectan dos o más líneas de transmisión. Cuando los datos llegan por una línea entrante, el elemento de conmutación debe elegir una línea saliente hacia la cual reenviarlos. En el pasado, estas computadoras de conmutación han recibido varios nombres; ahora se conocen como **enrutador**.

Aprovechemos el momento para hablar un poco sobre el término "subred". En un principio, su **único** significado era el de una colección de enrutadores y líneas de comunicación que transmitían paquetes desde el host de origen hasta el host de destino. Es necesario que nuestros lectores sepan que ha adquirido un segundo significado más reciente en conjunto con el direccionamiento de red. Hablaremos sobre este significado en el capítulo 5 y mientras nos apegaremos al significado original (una colección de líneas y enrutadores).

Según nuestra descripción de la WAN, ésta es muy parecida a una LAN alámbrica extensa, sólo que hay ciertas diferencias importantes que van más allá de los cables extensos. Por lo general, en una WAN los hosts y la subred pertenecen a distintas personas, quienes actúan también como operadores. En nuestro ejemplo, los empleados podrían ser responsables de sus propias computadoras mientras que el departamento de TI de la empresa está a cargo del resto de la red. En los siguientes ejemplos veremos límites más claros, en donde el proveedor de red o compañía telefónica opera la subred. Al separar los aspectos exclusivos de comunicación (la subred) de los aspectos relacionados con la aplicación (los hosts) se simplifica en forma considerable el diseño de la red en general.

Una segunda diferencia es que los enrutadores por lo general conectan distintos tipos de tecnología de red. Por ejemplo, las redes dentro de las oficinas pueden usar la tecnología de Ethernet conmutada mientras que las líneas de transmisión de larga distancia pueden ser enlaces SONET (que veremos en el capítulo 2). Se requiere algún dispositivo para conectarlas. El lector inteligente observará que esto va más allá de nuestra definición de una red. Esto significa que muchas redes WAN serán de hecho **interredes**, o redes compuestas formadas por más de una red. En la siguiente sección veremos más detalles sobre las interredes.

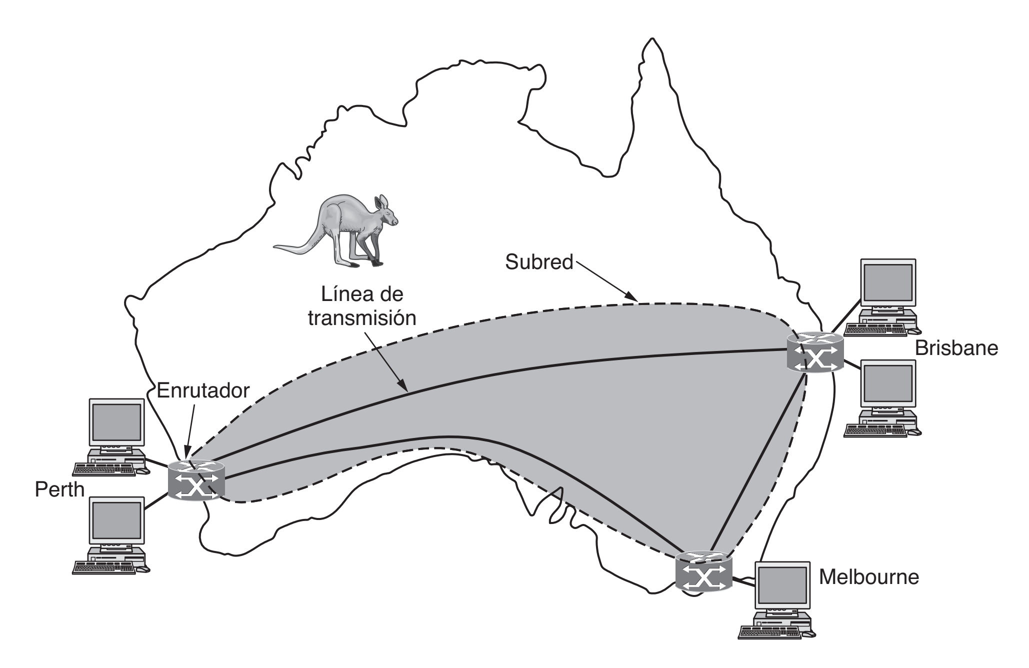


Figura 1-10. Una WAN que conecta tres sucursales en Australia.

Una última diferencia está en lo que se conecta a la subred. Podrían ser computadoras individuales, como en el caso de la conexión a redes LAN, o podrían ser redes LAN completas. Ésta es la forma en que se construyen redes más grandes a partir de otras más pequeñas. En lo que concierne a la subred, ésta hace el mismo trabajo.

Ahora estamos en posición de ver otras dos variedades de redes WAN. En primer lugar, en vez de rentar líneas de transmisión dedicadas, una empresa podría conectar sus oficinas a Internet. Esto le permite hacer conexiones entre las oficinas como enlaces virtuales que utilizan la capacidad subyacente de Internet. A este arreglo, que se muestra en la figura 1-11, se le denomina **VPN** (**Red Privada Virtual**, del inglés *Virtual Private Network*). Si se le compara con un arreglo dedicado, una VPN tiene la ventaja común de la virtualización, lo cual significa que provee flexibilidad en la reutilización de un recurso (conectividad a Internet). Para ver esto, considere lo fácil que sería conectar una cuarta oficina. Una VPN también tiene la desventaja común de la virtualización, lo cual significa que carece de control sobre los recursos subyacentes. Con una línea dedicada, la capacidad está clara. Con una VPN la capacidad puede variar según el servicio de Internet contratado.

La segunda variación es que una empresa distinta puede operar la subred. Al operador de la subred se le conoce como **proveedor de servicios de red** y las oficinas son sus clientes. En la figura 1-12 se muestra esta estructura. El operador de la subred se conecta también con otros clientes, siempre y cuando puedan pagar y les pueda proveer servicio. Como sería un servicio de red decepcionante si los clientes sólo pudieran enviarse paquetes entre sí, el operador de la subred también puede conectarse con otras redes que formen parte de Internet. A dicho operador de subred se le conoce como **ISP** (**Proveedor de Servicios de Internet**, del inglés *Internet Service Provider*) y la subred es una **red ISP**. Los clientes que se conectan al ISP reciben servicio de Internet.

Podemos usar la red ISP para ver por adelantado algunas cuestiones clave que estudiaremos en los capítulos posteriores. En la mayoría de las redes WAN, la red contiene muchas líneas de transmisión, cada una de las cuales conecta a un par de enrutadores. Si dos enrutadores que no comparten una línea de

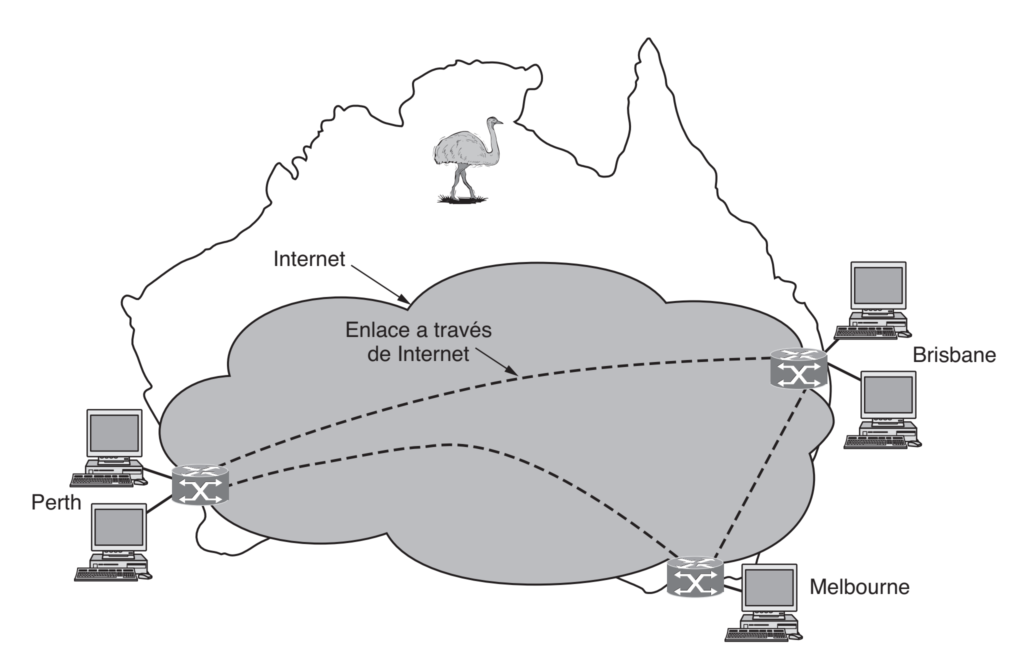


Figura 1-11. Una WAN que utiliza una red privada virtual.

transmisión desean comunicarse, deben hacerlo en forma indirecta a través de otros enrutadores. Puede haber muchas rutas en la red que conecten a estos dos enrutadores. Al proceso por el cual la red decide qué ruta tomar se le conoce como **algoritmo de enrutamiento**. Existen muchos algoritmos de este tipo. La manera en que cada enrutador toma la decisión de hacia dónde debe enviar el siguiente paquete se le denomina **algoritmo de reenvío**. También existen muchos de éstos. En el capítulo 5 estudiaremos ambos tipos de algoritmos con detalle.

Otros tipos de redes WAN utilizan mucho las tecnologías inalámbricas. En los sistemas de satélite, cada computadora en la Tierra tiene una antena a través de la cual es posible enviar y recibir datos de un satélite en órbita. Todas las computadoras pueden escuchar la salida *proveniente* del satélite y, en algunos casos, también pueden escuchar las transmisiones que envían sus computadoras vecinas *hacia* el satélite. Las redes de satélite son de difusión por naturaleza y son más útiles cuando es importante contar con la propiedad de difusión.

La red de telefonía celular es otro ejemplo de una WAN que utiliza tecnología inalámbrica. Este sistema ya pasó por tres generaciones y hay una cuarta por venir. La primera generación fue análoga y sólo para voz. La segunda fue digital y sólo para voz. La tercera generación es digital y se pueden transmitir tanto datos como voz. Cada estación base en un sistema celular cubre una distancia mucho mayor que una LAN inalámbrica, en donde el rango se mide en kilómetros en vez de decenas de metros. Las estaciones base se conectan entre sí mediante una red troncal que por lo general es alámbrica. Las velocidades de datos de las redes celulares se encuentran comúnmente en el orden de 1 Mbps, un valor mucho menor al de una LAN inalámbrica que puede estar en el orden de hasta 100 Mbps. En el capítulo 2 veremos muchos detalles sobre estas redes.

1.2.5 Interredes

Existen muchas redes en el mundo, a menudo con distintos componentes de hardware y software. Por lo general, las personas conectadas a una red se quieren comunicar con las personas conectadas a una red

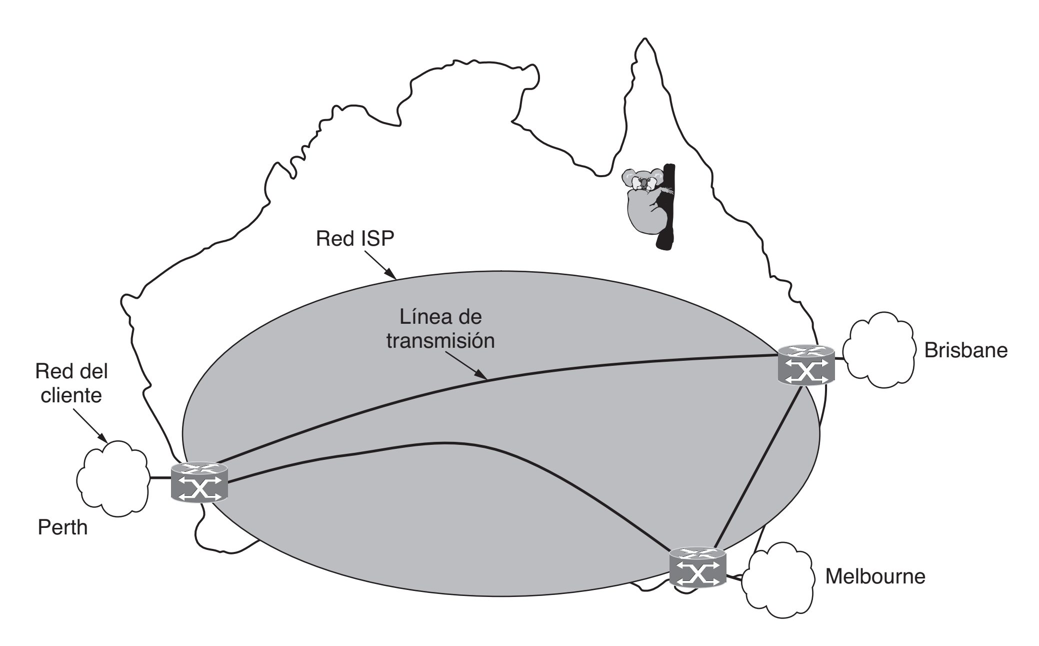


Figura 1-12. Una WAN que utiliza una red de ISP.

distinta; para lograrlo, es necesario conectar redes distintas que con frecuencia son incompatibles. A una colección de redes interconectadas se le conoce como **interred** o **internet**. Utilizaremos estos términos en un sentido genérico, en contraste a la red Internet mundial (que es una internet específica), a la cual nos referiremos siempre con I mayúscula. Internet usa redes de ISP para conectar redes empresariales, domésticas y muchos otros tipos más. Analizaremos la red Internet detalladamente más adelante.

A menudo se confunden las subredes, las redes y las interredes. El término "subred" tiene más sentido en el contexto de una red de área amplia, en donde se refiere a la colección de enrutadores y líneas de comunicación que pertenecen al operador de red. Como analogía, el sistema telefónico está compuesto por oficinas de conmutación telefónica conectadas entre sí mediante líneas de alta velocidad y conectadas a los hogares y negocios mediante líneas de baja velocidad. Estas líneas y equipos, que pertenecen y son administradas por la compañía telefónica, forman la subred del sistema telefónico. Los teléfonos en sí (los hosts en esta analogía) no forman parte de la subred.

Una red se forma al combinar una subred y sus hosts. Sin embargo, la palabra "red" a menudo también se utiliza en un sentido amplio. Podríamos describir una subred como una red, como en el caso de la "red ISP" de la figura 1-12. También podríamos describir una interred como una red, como en el caso de la WAN en la figura 1-10. Continuaremos con una práctica similar y cuando haya que diferenciar una red de otras distribuciones, nos apegaremos a nuestra definición original de una colección de computadoras interconectadas mediante una sola tecnología.

Ahora veamos detalladamente cómo está constituida una interred. Sabemos que una interred se forma cuando hay distintas redes interconectadas. A nuestro parecer, conectar una LAN y una WAN o conectar dos redes LAN es la forma usual de formar una interred, pero la industria no ha llegado a un buen acuerdo en cuanto a la terminología utilizada en esta área. Hay dos reglas prácticas y útiles a este respecto. En primer lugar, si varias organizaciones han pagado para construir distintas partes de la red y cada una se encarga de dar mantenimiento a la parte que le corresponde, entonces tenemos una interred en vez de una sola red. En segundo lugar, si la tecnología subyacente es distinta en diferentes partes (por ejemplo, difusión frente punto a punto y alámbrica frente a inalámbrica), es probable que sea una interred.

Para profundizar en este tema, hablaremos sobre la forma en que se pueden conectar dos redes distintas. El nombre general para una máquina que realiza una conexión entre dos o más redes y provee la traducción necesaria, tanto en términos de hardware como de software, es **puerta de enlace** (*gateway*). Las puertas de enlace se distinguen por la capa en la que operan en la jerarquía de protocolos. En la siguiente sección hablaremos mucho más sobre las capas y las jerarquías de protocolos, pero por ahora basta con imaginar que las capas superiores están más relacionadas con las aplicaciones (como la web), mientras que las capas inferiores están más relacionadas con los enlaces de transmisión (como Ethernet).

Como el beneficio de formar una internet es para conectar computadoras entre distintas redes, no es conveniente usar una puerta de enlace de una capa demasiado baja, ya que no podremos realizar conexiones entre distintos tipos de redes. Tampoco es conveniente usar una puerta de enlace de una capa demasiado alta, o de lo contrario la conexión sólo funcionará para ciertas aplicaciones. A la capa en la parte media que resulta ser la "ideal" se le denomina comúnmente **capa de red**; un enrutador es una puerta de enlace que conmuta paquetes en la capa de red. Así, para detectar una interred o internet hay que buscar una red que tenga enrutadores.

1.3 SOFTWARE DE RED

Las primeras redes de computadoras se diseñaron teniendo en cuenta al hardware como punto principal y al software como secundario. Pero esta estrategia ya no funciona. Ahora el software de red está muy estructurado. En las siguientes secciones examinaremos con cierto detalle la técnica para estructurar el software. La metodología aquí descrita constituye la piedra angular de todo el libro y, por lo tanto, se repetirá en secciones posteriores.

1.3.1 Jerarquías de protocolos

Para reducir la complejidad de su diseño, la mayoría de las redes se organizan como una pila de **capas** o **niveles**, cada una construida a partir de la que está abajo. El número de capas, su nombre, el contenido de cada una y su función difieren de una red a otra. El propósito de cada capa es ofrecer ciertos servicios a las capas superiores, mientras les oculta los detalles relacionados con la forma en que se implementan los servicios ofrecidos. Es decir, cada capa es un tipo de máquina virtual que ofrece ciertos servicios a la capa que está encima de ella.

En realidad este concepto es familiar y se utiliza en muchas áreas de las ciencias computacionales, en donde se le conoce de muchas formas: ocultamiento de información, tipos de datos abstractos, encapsulamiento de datos y programación orientada a objetos. La idea fundamental es que una pieza particular de software (o hardware) provee un servicio a sus usuarios pero mantiene ocultos los detalles de su estado interno y los algoritmos que utiliza.

Cuando la capa *n* en una máquina lleva a cabo una conversación con la capa *n* en otra máquina, a las reglas y convenciones utilizadas en esta conversación se les conoce como el protocolo de la capa *n*. En esencia, un **protocolo** es un acuerdo entre las partes que se comunican para establecer la forma en que se llevará a cabo esa comunicación. Como analogía, cuando a un hombre le presentan una mujer, ella puede elegir si extiende su mano o no. Él a su vez, puede decidir entre estrechar la mano o besarla, dependiendo por ejemplo de si ella es una abogada estadounidense en una reunión de negocios, o una princesa europea en un baile formal. Si se viola el protocolo se hará más difícil la comunicación, si no es que se vuelve imposible.

En la figura 1-13 se ilustra una red de cinco capas. Las entidades que conforman las correspondientes capas en diferentes máquinas se llaman **iguales** (*peers*). Los iguales pueden ser procesos de software,

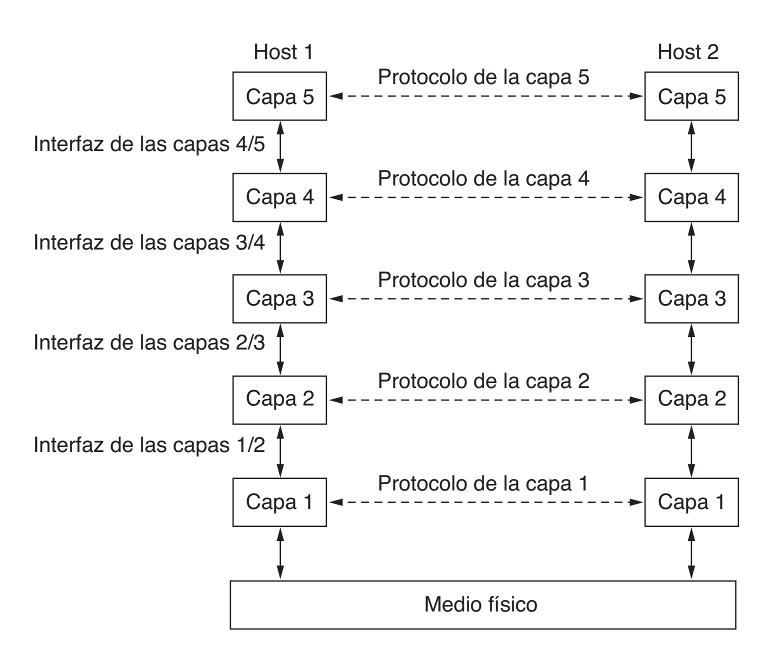


Figura 1-13. Capas, protocolos e interfaces.

dispositivos de hardware o incluso seres humanos. En otras palabras, los iguales son los que se comunican a través del protocolo.

En realidad no se transfieren datos de manera directa desde la capa *n* de una máquina a la capa *n* de otra máquina, sino que cada capa pasa los datos y la información de control a la capa inmediatamente inferior, hasta que se alcanza a la capa más baja. Debajo de la capa 1 se encuentra el **medio físico** a través del cual ocurre la comunicación real. En la figura 1-13 se muestra la comunicación virtual con líneas punteadas y la comunicación física con líneas sólidas.

Entre cada par de capas adyacentes hay una **interfaz**. Ésta define las operaciones y servicios primitivos que pone la capa más baja a disposición de la capa superior inmediata. Cuando los diseñadores de redes deciden cuántas capas incluir en una red y qué debe hacer cada una, la consideración más importante es definir interfaces limpias entre las capas. Al hacer esto es necesario que la capa desempeñe un conjunto específico de funciones bien entendidas. Además de minimizar la cantidad de información que se debe pasar entre las capas, las interfaces bien definidas también simplifican el reemplazo de una capa con un protocolo o implementación totalmente diferente (por ejemplo, reemplazar todas las líneas telefónicas por canales de satélite), ya que todo lo que se requiere del nuevo protocolo o implementación es que ofrezca exactamente el mismo conjunto de servicios a su vecino de arriba, como lo hacía el protocolo o la implementación anterior. Es común que distintos hosts utilicen diferentes implementaciones del mismo protocolo (a menudo escrito por otras compañías). De hecho, el protocolo en sí puede cambiar en cierta capa sin que las capas superior e inferior lo noten.

A un conjunto de capas y protocolos se le conoce como **arquitectura de red**. La especificación de una arquitectura debe contener suficiente información como para permitir que un programador escriba el programa o construya el hardware para cada capa, de manera que se cumpla correctamente el protocolo apropiado. Ni los detalles de la implementación ni la especificación de las interfaces forman parte de la arquitectura, ya que están ocultas dentro de las máquinas y no se pueden ver desde el exterior. Ni siquiera es necesario que las interfaces en todas las máquinas de una red sean iguales, siempre y cuando cada máquina pueda utilizar todos los protocolos correctamente. La lista de los protocolos utilizados por cierto

sistema, un protocolo por capa, se le conoce como **pila de protocolos**. Las arquitecturas de red, las pilas de protocolos y los protocolos mismos son los temas principales de este libro.

Una analogía podría ayudar a explicar la idea de la comunicación entre múltiples capas. Imagine a dos filósofos (procesos de iguales en la capa 3), uno de los cuales habla urdú e inglés, mientras que el otro habla chino y francés. Como no tienen un lenguaje común, cada uno contrata a un traductor (procesos de iguales en la capa 2) y cada uno de los traductores a su vez contacta a una secretaria (procesos de iguales en la capa 1). El filósofo 1 desea comunicar su afición por el *oryctolagus cuniculus* a su igual. Para ello pasa un mensaje (en español) a través de la interfaz de las capas 2-3 a su traductor para decirle: "Me gustan los conejos", como se muestra en la figura 1-14. Los traductores han acordado un idioma neutral conocido por ambos, el holandés, así el mensaje es convertido a "*Ik vind konijnen leuk*". La elección del idioma es el protocolo de la capa 2 y depende de los procesos de iguales de dicha capa.

Después, el traductor pasa el mensaje a una secretaria para que lo transmita, por ejemplo, mediante correo electrónico (el protocolo de la capa 1). Cuando el mensaje llega a la otra secretaria, ésta lo pasa al traductor local, quien lo traduce al francés y lo pasa a través de la interfaz de las capas 2-3 al segundo filósofo 2. Observe que cada protocolo es totalmente independiente de los demás siempre y cuando no cambien las interfaces. Por ejemplo, los traductores pueden cambiar de holandés al finlandés siempre y cuando ambos estén de acuerdo y ninguno cambie su interfaz con las capas 1 o 3. De manera similar, las secretarias pueden cambiar del correo electrónico al teléfono sin molestar (o incluso informar) a las demás capas. Cada proceso puede agregar algo de información destinada sólo a su igual. Esta información no se pasa a la capa superior.

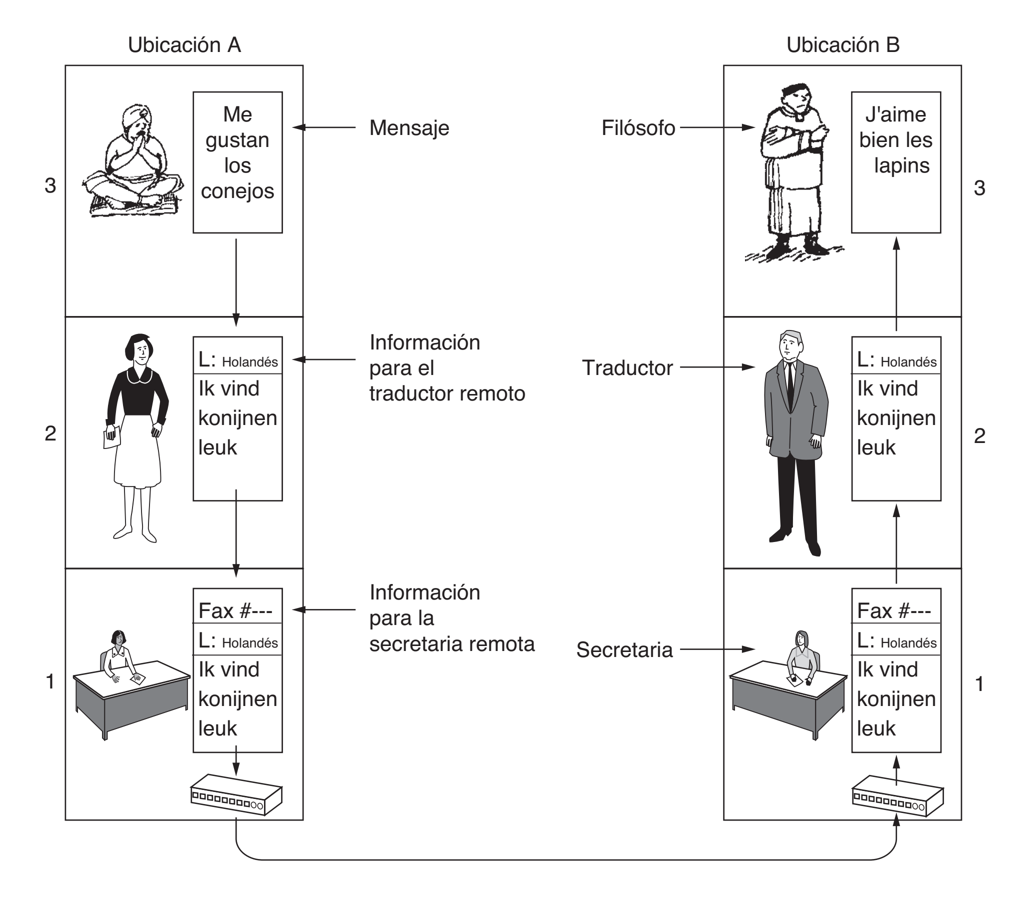


Figura 1-14. La arquitectura filósofo-traductor-secretaria.

Ahora considere un ejemplo más técnico: cómo proveer comunicación a la capa superior de la red de cinco capas de la figura 1-15. Un proceso de aplicación que se ejecuta en la capa 5 produce un mensaje, *M*, y lo pasa a la capa 4 para que lo transmita. La capa 4 coloca un **encabezado** al frente del mensaje para identificarlo y pasa el resultado a la capa 3. El encabezado incluye información de control, como direcciones, para permitir que la capa 4 en la máquina de destino entregue el mensaje. Otros ejemplos de la información de control que se utiliza en algunas capas son los números de secuencia (en caso de que la capa inferior no preserve el orden del mensaje), los tamaños y los tiempos.

En muchas redes no se impone un límite en cuanto al tamaño de los mensajes que se transmiten en el protocolo de la capa 4, pero casi siempre hay un límite impuesto por el protocolo de la capa 3. En consecuencia, la capa 3 debe descomponer los mensajes entrantes en unidades más pequeñas llamadas paquetes, y colocar un encabezado al frente de cada paquete. En este ejemplo, *M* se divide en dos partes: M_1 y M_2, los cuales se transmitirán por separado.

La capa 3 decide cuál de las líneas salientes usar y pasa los paquetes a la capa 2; esta última agrega a cada pieza no sólo un encabezado, sino también un terminador, y pasa la unidad restante a la capa 1 para su transmisión física. En la máquina receptora el mensaje pasa hacia arriba, de capa en capa, y los encabezados se van eliminando a medida que progresa. Ninguno de los encabezados para las capas inferiores a *n* se pasa a la capa *n*.

Lo importante a entender sobre la figura 1-15 es la relación entre la comunicación virtual y real, además de la diferencia entre los protocolos y las interfaces. Por ejemplo, los procesos de iguales en la capa 4 piensan conceptualmente en su comunicación como si fuera “horizontal” y utilizan el protocolo de la capa 4. Es probable que cada uno tenga procedimientos llamados *EnviarAlOtroLado* y *RecibirDelOtroLado*, aun cuando en realidad estos procedimientos se comunican con las capas inferiores a través de la interfaz de las capas 3-4, no con el otro lado.

La abstracción de los procesos de iguales es imprescindible para todo diseño de red. Al usarla, la inmanejable tarea de diseñar toda la red se puede fragmentar en varios problemas de diseño más pequeños y manejables, es decir, el diseño de las capas individuales.

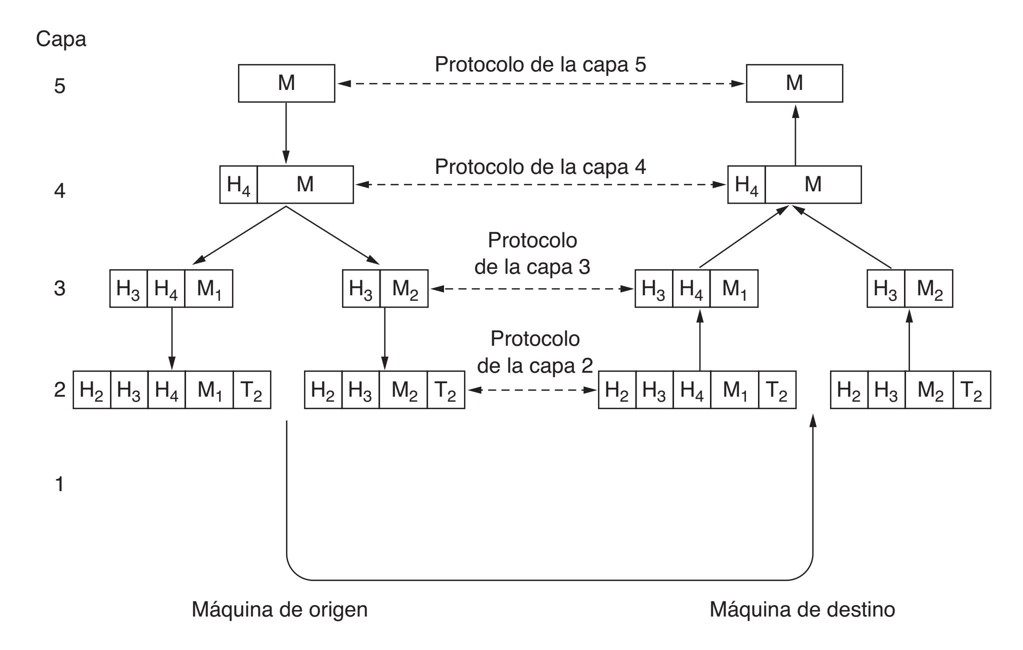


Figura 1-15. Ejemplo de flujo de información que soporta la comunicación virtual en la capa 5.

Aunque la sección 1.3 se llama "Software de red", vale la pena mencionar que las capas inferiores de una jerarquía de protocolos se implementan con frecuencia en el hardware o firmware. Sin embargo, están implicados los algoritmos de protocolos complejos, incluso aunque estén integrados (en todo o en parte) al hardware.

1.3.2 Aspectos de diseño para las capas

Algunos de los aspectos clave de diseño que ocurren en las redes de computadoras están presentes en las diversas capas. A continuación mencionaremos brevemente los más importantes.

La confiabilidad es el aspecto de diseño enfocado en verificar que una red opere correctamente, aun cuando esté formada por una colección de componentes que sean, por sí mismos, poco confiables. Piense en los bits de un paquete que viajan a través de la red. Existe la posibilidad de que algunas de estas piezas se reciban dañadas (invertidas) debido al ruido eléctrico, a las señales aleatorias inalámbricas, a fallas en el hardware, a errores del software, etc. ¿Cómo es posible detectar y corregir estos errores?

Un mecanismo para detectar errores en la información recibida utiliza códigos de **detección de errores**. Así, la información que se recibe de manera incorrecta puede retransmitirse hasta que se reciba de manera correcta. Los códigos más poderosos cuentan con **corrección de errores**, en donde el mensaje correcto se recupera a partir de los bits posiblemente incorrectos que se recibieron originalmente. Ambos mecanismos funcionan añadiendo información redundante. Se utilizan en capas bajas para proteger los paquetes que se envían a través de enlaces individuales, y en capas altas para verificar que el contenido correcto fue recibido.

Otro aspecto de la confiabilidad consiste en encontrar una ruta funcional a través de una red. A menudo hay múltiples rutas entre origen y destino, y en una red extensa puede haber algunos enlaces o enrutadores descompuestos. Suponga que la red está caída en Alemania. Los paquetes que se envían de Londres a Roma a través de Alemania no podrán pasar, pero para evitar esto, podríamos enviar los paquetes de Londres a Roma vía París. La red debería tomar esta decisión de manera automática. A este tema se le conoce como **enrutamiento**.

Un segundo aspecto de diseño se refiere a la evolución de la red. Con el tiempo, las redes aumentan su tamaño y emergen nuevos diseños que necesitan conectarse a la red existente. Recientemente vimos el mecanismo de estructuración clave que se utiliza para soportar el cambio dividiendo el problema general y ocultando los detalles de la implementación: **distribución de protocolos en capas**. También existen muchas otras estrategias.

Como hay muchas computadoras en la red, cada capa necesita un mecanismo para identificar los emisores y receptores involucrados en un mensaje específico. Este mecanismo se conoce como **direccionamiento** o **nombramiento** en las capas altas y bajas, respectivamente.

Un aspecto del crecimiento es que las distintas tecnologías de red a menudo tienen diferentes limitaciones. Por ejemplo, no todos los canales de comunicación preservan el orden de los mensajes que se envían en ellos, por lo cual es necesario idear soluciones para enumerar los mensajes. Otro ejemplo es el de las diferencias en el tamaño máximo de un mensaje que las redes pueden transmitir. Esto provoca la creación de mecanismos para desensamblar, transmitir y después volver a ensamblar los mensajes. A este tema en general se le conoce como **interconexión de redes** (*internetworking*).

Cuando las redes crecen, surgen nuevos problemas. Las ciudades pueden tener problemas de tráfico, escasez de números telefónicos y es fácil perderse. No muchas personas tienen estos problemas en su propio vecindario, pero en toda la ciudad pueden representar un gran problema. Se dice que los diseños que siguen funcionando bien cuando la red aumenta su tamaño son **escalables**.

Un tercer aspecto de diseño radica en la asignación de recursos. Las redes proveen un servicio a los hosts desde sus recursos subyacentes, como la capacidad de las líneas de transmisión. Para hacer bien su

trabajo necesitan mecanismos que dividan sus recursos de manera que un host no interfiera demasiado con otro host.

Muchos diseños comparten el ancho de banda de una red en forma dinámica, de acuerdo con las necesidades a corto plazo de los hosts, en vez de otorgar a cada host una fracción fija del ancho de banda que puede llegar a utilizar o quizás no. A este diseño se le denomina **multiplexado estadístico**, lo cual significa que se comparten los recursos con base en la demanda. Se puede aplicar en capas bajas para un solo enlace o en capas altas para una red, o incluso para aplicaciones que utilizan la red.

Un problema de asignación que ocurre en todas las capas es cómo evitar que un emisor rápido inunde de datos a un receptor lento. Con frecuencia se utiliza retroalimentación del receptor al emisor. A este tema se le denomina **control de flujo**. Algunas veces el problema es que la red sufre un exceso de solicitudes debido a que hay demasiadas computadoras que desean enviar una gran cantidad de información y la red no lo puede entregar todo. A esta sobrecarga de la red se le conoce como **congestión**. Una estrategia es que cada computadora reduzca su demanda cuando experimenta congestión. Esto también se puede usar en todas las capas.

Es interesante observar que la red puede ofrecer más recursos que simplemente el ancho de banda. Para usos como transmitir video en vivo, la puntualidad de la entrega es en extremo importante. La mayoría de las redes deben proveer servicio a las aplicaciones que desean esta entrega en **tiempo real** al mismo tiempo que proveen servicio a las aplicaciones que desean un alto rendimiento. La **calidad del servicio** es el nombre que se da a los mecanismos que reconcilian estas demandas competitivas.

El último aspecto de diseño importante es asegurar la red y defenderla contra distintos tipos de amenazas. Una de las amenazas que mencionamos antes es la de espiar las comunicaciones. Los mecanismos que proveen **confidencialidad** nos defienden contra esta amenaza y se utilizan en múltiples capas. Los mecanismos de **autenticación** evitan que alguien se haga pasar por otra persona. Se pueden usar para diferenciar los sitios web bancarios falsos de los verdaderos, o para permitir que la red celular verifique que una llamada realmente provenga de nuestro teléfono para pagar la cuenta. Otros mecanismos para la **integridad** evitan cambios clandestinos a los mensajes, como cuando se altera el mensaje "cargar $10 a mi cuenta" para convertirlo en "cargar $1000 dólares a mi cuenta". Todos estos diseños se basan en la criptografía que estudiaremos en el capítulo 8.

1.3.3 Comparación entre servicio orientado a conexión y servicio sin conexión

Las capas pueden ofrecer dos tipos distintos de servicio a las capas superiores: orientado a conexión y sin conexión. En esta sección analizaremos estos dos tipos y examinaremos las diferencias entre ellos.

El servicio **orientado a conexión** está modelado a partir del sistema telefónico. Para hablar con alguien levantamos el auricular, marcamos el número, hablamos y después colgamos. De manera similar, para usar un servicio de red orientado a conexión, el usuario del servicio establece primero una conexión, la utiliza y después la libera. El aspecto esencial de una conexión es que funciona como un tubo: el emisor mete objetos (bits) en un extremo y el receptor los toma en el otro extremo. En la mayoría de los casos se conserva el orden de manera que los bits llegan en el orden en el que se enviaron.

En algunos casos al establecer una conexión, el emisor, el receptor y la subred llevan a cabo una **negociación** en cuanto a los parámetros que se van a usar, como el tamaño máximo del mensaje, la calidad requerida del servicio y demás cuestiones relacionadas. Por lo general, uno de los lados hace una propuesta y el otro puede aceptarla, rechazarla o elaborar una contrapropuesta. Un circuito es otro nombre para una conexión con recursos asociados, como un ancho de banda fijo. Esto se remonta a la red telefónica, en la cual un circuito era una ruta sobre alambre que transmitía una conversación telefónica.

En contraste al servicio orientado a la conexión, el servicio **sin conexión** está modelado a partir del sistema postal. Cada mensaje (carta) lleva la dirección de destino completa, y cada uno es enrutado hacia

los nodos intermedios dentro del sistema, en forma independiente a todos los mensajes subsecuentes. Hay distintos nombres para los mensajes en diferentes contextos: un **paquete** es un mensaje en la capa de red. Cuando los nodos intermedios reciben un mensaje completo antes de enviarlo al siguiente nodo, se le llama **conmutación de almacenamiento y envío**. La alternativa en donde la transmisión subsiguiente de un mensaje en un nodo empieza antes de que éste la reciba por completo, se conoce como "conmutación al vuelo". Por lo general, cuando se envían dos mensajes al mismo destino, el primero que se envíe será el primero en llegar. Sin embargo, es posible que el primero que se envié se retrase de manera que el segundo llegue primero.

Cada tipo de servicio se puede caracterizar con base en su confiabilidad. Algunos servicios son confiables en cuanto a que nunca pierden datos. Por lo general, para implementar un servicio confiable, el receptor tiene que confirmar la recepción de cada mensaje, de manera que el emisor esté seguro de que hayan llegado. El proceso de confirmación de recepción introduce sobrecarga y retardos, que a menudo valen la pena pero algunas veces no son deseables.

Una situación común en la que es apropiado un servicio confiable orientado a la conexión es la transferencia de archivos. El propietario del archivo desea estar seguro de que todos los bits lleguen correctamente y en el mismo orden en el que se enviaron. Muy pocos clientes que transfieren archivos preferirían un servicio que ocasionalmente revuelva o pierda unos cuantos bits, incluso aunque fuera mucho más rápido.

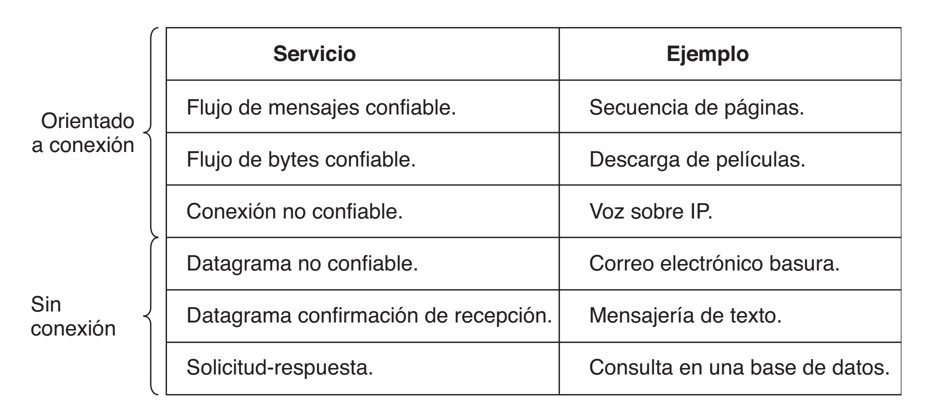
El servicio confiable orientado a la conexión tiene dos variaciones menores: secuencias de mensajes y flujos de bytes. En la primera variante se conservan los límites de los mensajes. Cuando se envían dos mensajes de 1024 bytes, llegan como dos mensajes distintos de 1024 bytes y nunca como un mensaje de 2048 bytes. En la segunda variante, la conexión es simplemente un flujo de bytes sin límites en los mensajes. Cuando llegan 2048 bytes al receptor, no hay manera de saber si se enviaron como un mensaje de 2048 bytes, como dos mensajes de 1024 bytes o como 2048 mensajes de 1 byte. Si se envían las páginas de un libro a través de una red a una máquina de fotocomposición en forma de mensajes separados, probablemente sea importante preservar los límites de los mensajes. Por otro lado, para descargar una película en DVD, todo lo que se necesita es un flujo de bytes del servidor a la computadora del usuario. Los límites de los mensajes dentro de la película no son relevantes.

En algunas aplicaciones, los retardos de tránsito ocasionados por las confirmaciones de recepción son inaceptables. Una de estas aplicaciones es el tráfico de voz digitalizada o **voz sobre IP**. Es preferible para los usuarios del teléfono escuchar un poco de ruido en la línea de vez en cuando que experimentar un retardo al esperar las confirmaciones de recepción. De manera similar, al transmitir una conferencia de video no hay problema si unos cuantos píxeles están mal, pero es molesto cuando la imagen se sacude mientras el flujo se detiene y avanza para corregir errores.

No todas las aplicaciones requieren conexiones. Por ejemplo, los emisores de correo electrónico basura (*spammers*) envían su correo a muchos destinatarios. Es probable que el emisor no quiera tener que pasar por el problema de establecer y desmantelar una conexión con un destinatario sólo para enviarle un mensaje. Tampoco es esencial una entrega cien por ciento confiable, sobre todo si eso es más costoso. Todo lo que se requiere es una forma de enviar un solo mensaje que tenga una muy alta probabilidad de llegar, aunque sin garantías. Al servicio sin conexión no confiable (que significa sin confirmación de recepción) se le denomina servicio de **datagramas**, en analogía al servicio de telegramas que tampoco devuelve una confirmación de recepción al emisor. A pesar de ser poco confiable, es la forma más dominante en la mayoría de las redes por motivos que veremos más adelante.

En otros casos es conveniente no tener que establecer una conexión para enviar un mensaje, pero la confiabilidad es esencial. En estas aplicaciones se puede utilizar el servicio de **datagramas con confirmación de recepción**. Es como enviar una carta certificada y solicitar una confirmación de recepción. Al regresar la confirmación de recepción el emisor tiene la absoluta certeza de que la carta se entregó al destinatario correcto y que no se perdió en el camino. La mensajería de texto en los teléfonos móviles es un ejemplo.

Hay otro servicio conocido como servicio de **solicitud-respuesta**. En este servicio el emisor transmite un solo datagrama que contiene una solicitud; al receptor envía la respuesta. El servicio de solicitud-respuesta se utiliza mucho para implementar la comunicación en el modelo cliente-servidor; el cliente emite una petición y el servidor le responde. Por ejemplo, el cliente de un teléfono móvil podría enviar una consulta a un servidor de mapas para recuperar los datos del mapa de la ubicación actual. En la figura 1-16 se sintetizan los tipos de servicios antes descritos.



	Servicio	Ejemplo
Orientado a conexión	Flujo de mensajes confiable.	Secuencia de páginas.
	Flujo de bytes confiable.	Descarga de películas.
	Conexión no confiable.	Voz sobre IP.
Sin conexión	Datagrama no confiable.	Correo electrónico basura.
	Datagrama confirmación de recepción.	Mensajería de texto.
	Solicitud-respuesta.	Consulta en una base de datos.

Figura 1-16. Seis tipos distintos de servicios.

Tal vez el concepto de usar una comunicación poco confiable le parezca confuso en un principio. Después de todo, ¿por qué preferiría alguien una comunicación poco confiable en vez de una comunicación confiable? Primero que nada, tal vez la comunicación confiable (en nuestro contexto significa que es con confirmación de recepción) no esté disponible en cierta capa. Por ejemplo, Ethernet no provee una comunicación confiable. Los paquetes se pueden dañar ocasionalmente durante el tránsito. Las capas de protocolos más altas deben tener la capacidad de recuperarse de este problema. En particular, muchos servicios confiables se basan en un servicio de datagramas no confiables. En segundo lugar, los retardos inherentes al proveer un servicio confiable tal vez sean inaceptables, en especial en las aplicaciones de tiempo real como multimedia. Éstas son las razones por las que coexisten la comunicación confiable y la comunicación poco confiable.

1.3.4 Primitivas de servicios

Un servicio se puede especificar de manera formal como un conjunto de **primitivas** (operaciones) disponibles a los procesos de usuario para que accedan al servicio. Estas primitivas le indican al servicio que desarrollen alguna acción o que informen sobre la acción que haya tomado una entidad par. Si la pila de protocolos se encuentra en el sistema operativo, como se da en la mayoría de los casos, por lo general las primitivas son llamadas al sistema. Estas llamadas provocan un salto al modo de kernel, que a su vez devuelve el control de la máquina al sistema operativo para que envíe los paquetes necesarios.

El conjunto de primitivas disponibles depende de la naturaleza del servicio que se va a ofrecer. Las primitivas para el servicio orientado a conexión son distintas de las primitivas para el servicio sin conexión. Como un ejemplo mínimo de las primitivas de servicio que se podrían ofrecer para implementar un flujo de bytes confiable, considere las primitivas que se enlistan en la figura 1-17. Estas primitivas serán familiares para los fanáticos de la interfaz de sockets de Berkeley, ya que son una versión simplificada de esa interfaz.

Primitiva	Significado
LISTEN	Bloquea en espera de una conexión entrante.
CONNECT	Establece una conexión con un igual en espera.
ACCEPT	Acepta una conexión entrante de un igual.
RECEIVE	Bloquea en espera de un mensaje entrante.
SEND	Envía un mensaje al igual.
DISCONNECT	Termina una conexión.

Figura 1-17. Seis primitivas de servicios que proveen un servicio simple orientado a conexión.

Podríamos usar estas primitivas para una interacción petición-respuesta en un entorno cliente-servidor. Para ilustrar esto, vamos a esbozar un protocolo simple que implementa el servicio mediante datagramas con confirmación de recepción.

Primero, el servidor ejecuta LISTEN para indicar que está preparado para aceptar conexiones entrantes. Una forma común de implementar LISTEN es mediante una llamada de bloqueo del sistema. Después de ejecutar la primitiva, el proceso servidor se bloquea hasta que aparezca una petición de conexión.

Después, el proceso cliente ejecuta CONNECT para establecer una conexión con el servidor. La llamada a CONNECT necesita especificar con quién se va a realizar la conexión, por lo que podría incluir un parámetro para proporcionar la dirección del servidor. A continuación, lo más común es que el sistema operativo envíe un paquete al igual para pedirle que se conecte, como se muestra en la sección (1) de la figura 1-18. El proceso cliente se suspende hasta que haya una respuesta.

Cuando el paquete llega al servidor, el sistema operativo ve que el paquete solicita una conexión. Verifica que haya alguien escuchando y, en ese caso, desbloquea al que está escuchando. Ahora el proceso servidor puede establecer la conexión con la llamada a ACCEPT. Esta llamada envía una respuesta (2) de vuelta al proceso cliente para aceptar la conexión. Al llegar esta respuesta se libera el cliente. En este punto, el cliente y el servidor se están ejecutando y tienen una conexión establecida.

La analogía obvia entre este protocolo y la vida real es un cliente que llama al gerente de servicio al cliente de una empresa. Al empezar el día, el gerente de servicio se sienta a un lado de su teléfono en caso de que suene. Después, un cliente hace una llamada. Cuando el gerente levanta el teléfono se establece la conexión.

El siguiente paso es que el servidor ejecute RECEIVE y se prepare para aceptar la primera petición. Por lo general, el servidor hace esto justo después de ser liberado de la primitiva LISTEN, antes de que la confirmación de recepción pueda regresar al cliente. La llamada a RECEIVE bloquea al servidor.

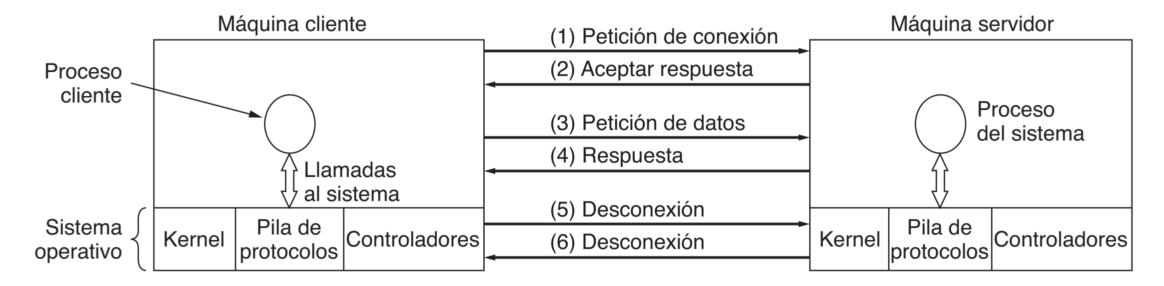


Figura 1-18. Una interacción cliente-servidor simple mediante el uso de datagramas con confirmación de recepción.

Entonces, el cliente ejecuta SEND para transmitir su petición (3) después de ejecutar RECEIVE para obtener la respuesta. La llegada del paquete solicitado a la máquina servidor desbloquea el servidor, de manera que pueda manejar la petición. Después de realizar su trabajo, el servidor usa SEND para devolver la respuesta al cliente (4). Al llegar este paquete se desbloquea el cliente, que ahora puede inspeccionar la respuesta. Si el cliente tiene peticiones adicionales, puede hacerlas ahora.

Cuando el cliente termina, ejecuta DISCONNECT para terminar la conexión (5). Por lo general una primitiva DISCONNECT inicial es una llamada de bloqueo, la cual suspende al cliente y envía un paquete al servidor para indicar que ya no necesita la conexión. Cuando el servidor recibe el paquete también emite una primitiva DISCONNECT por su cuenta, envía una confirmación de recepción al cliente y libera la conexión (6). Cuando el paquete del servidor regresa a la máquina cliente, se libera el proceso cliente y se interrumpe la conexión. En esencia, así es como funciona la comunicación orientada a conexión.

Por desgracia la vida no es tan simple. Aquí pueden salir mal muchas cosas. La sincronización puede estar mal (por ejemplo, que termine CONNECT antes de LISTEN), se pueden perder paquetes, etc. Más adelante analizaremos con mayor detalle estas cuestiones, pero por el momento en la figura 1-18 se resume la forma en que podría trabajar la comunicación cliente-servidor mediante datagramas con confirmación de recepción para poder ignorar los paquetes perdidos.

Dado que se requieren seis paquetes para completar este protocolo, tal vez se pregunte por qué no utilizar mejor un protocolo sin conexión. La respuesta es que en un mundo perfecto podría ser así, en cuyo caso sólo se necesitarían dos paquetes: uno para la petición y otro para la respuesta. Pero cuando hay mensajes extensos en cualquier dirección (por ejemplo, un archivo de un megabyte), errores de transmisión y paquetes perdidos, la situación cambia. Si la respuesta consistiera de cientos de paquetes, algunos de los cuales se pudieran perder durante la transmisión, ¿cómo sabría el cliente que faltan algunas piezas?, ¿cómo sabría si el último paquete que se recibió fue en realidad el último paquete enviado? Suponga que el cliente desea un segundo archivo. ¿Cómo podría diferenciar el paquete 1 del segundo archivo de un paquete 1 perdido del primer archivo que por fin pudo llegar al cliente? En resumen, en el mundo real es inadecuado usar un protocolo simple de petición-respuesta a través de una red poco confiable. En el capítulo 3 estudiaremos con detalle una variedad de protocolos que solucionan éstos y otros problemas. Por el momento basta con decir que algunas veces es conveniente tener un flujo de bytes ordenado y confiable entre procesos.

1.3.5 La relación entre servicios y protocolos

Los servicios y los protocolos son conceptos distintos. Esta distinción es tan importante que la enfatizaremos una vez más. Un *servicio* es un conjunto de primitivas (operaciones) que una capa proporciona a la capa que está encima de ella. El servicio define qué operaciones puede realizar la capa en beneficio de sus usuarios, pero no dice nada sobre cómo se implementan estas operaciones. Un servicio se relaciona con una interfaz entre dos capas, en donde la capa inferior es el proveedor del servicio y la capa superior es el usuario.

En contraste, un *protocolo* es un conjunto de reglas que rigen el formato y el significado de los paquetes o mensajes que intercambian las entidades iguales en una capa. Las entidades utilizan protocolos para implementar sus definiciones de servicios. Pueden cambiar sus protocolos a voluntad, siempre y cuando no cambien el servicio visible para sus usuarios. De esta manera, el servicio y el protocolo no dependen uno del otro. Éste es un concepto clave que cualquier diseñador de red debe comprender bien.

Para repetir este punto importante, los servicios se relacionan con las interfaces entre capas, como se muestra en la figura 1-19. En contraste, los protocolos se relacionan con los paquetes que se envían entre las entidades pares de distintas máquinas. Es muy importante no confundir los dos conceptos.

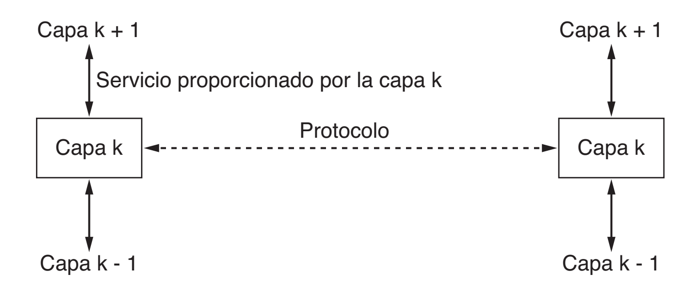


Figura 1-19. La relación entre un servicio y un protocolo.

Vale la pena mencionar una analogía con los lenguajes de programación. Un servicio es como un tipo de datos abstracto o un objeto en un lenguaje orientado a objetos. Define las operaciones que se pueden realizar en un objeto, pero no especifica cómo se implementan estas operaciones. En contraste, un protocolo se relaciona con la *implementación* del servicio y como tal, no es visible al usuario del mismo.

Muchos protocolos antiguos no diferenciaban el servicio del protocolo. En efecto, una capa típica podría tener una primitiva de servicio SEND PACKET en donde el usuario proporcionaba un apuntador hacia un paquete completamente ensamblado. Este arreglo significaba que los usuarios podían ver de inmediato todos los cambios en el protocolo. Ahora, la mayoría de los diseñadores de redes consideran dicho diseño como un error garrafal.

1.4 MODELOS DE REFERENCIA

Ahora que hemos analizado en lo abstracto las redes basadas en capas, es tiempo de ver algunos ejemplos. Analizaremos dos arquitecturas de redes importantes: el modelo de referencia OSI y el modelo de referencia TCP/IP. Aunque ya casi no se utilizan los *protocolos* asociados con el modelo OSI, el *modelo* en sí es bastante general y sigue siendo válido; asimismo, las características en cada nivel siguen siendo muy importantes. El modelo TCP/IP tiene las propiedades opuestas: el modelo en sí no se utiliza mucho, pero los protocolos son usados ampliamente. Por esta razón veremos ambos elementos con detalle. Además, algunas veces podemos aprender más de los fracasos que de los éxitos.

1.4.1 El modelo de referencia OSI

El modelo OSI se muestra en la figura 1-20 (sin el medio físico). Este modelo se basa en una propuesta desarrollada por la Organización Internacional de Normas (ISO) como el primer paso hacia la estandarización internacional de los protocolos utilizados en las diversas capas (Day y Zimmerman, 1983). Este modelo se revisó en 1995 (Day, 1995) y se le llama **Modelo de referencia OSI** (**Interconexión de Sistemas Abiertos**, del inglés *Open Systems Interconnection*) de la ISO puesto que se ocupa de la conexión de sistemas abiertos; esto es, sistemas que están abiertos a la comunicación con otros sistemas. Para abreviar, lo llamaremos **modelo OSI**.

El modelo OSI tiene siete capas. Los principios que se aplicaron para llegar a las siete capas se pueden resumir de la siguiente manera:

1. Se debe crear una capa en donde se requiera un nivel diferente de abstracción.
2. Cada capa debe realizar una función bien definida.
3. La función de cada capa se debe elegir teniendo en cuenta la definición de protocolos estandarizados internacionalmente.

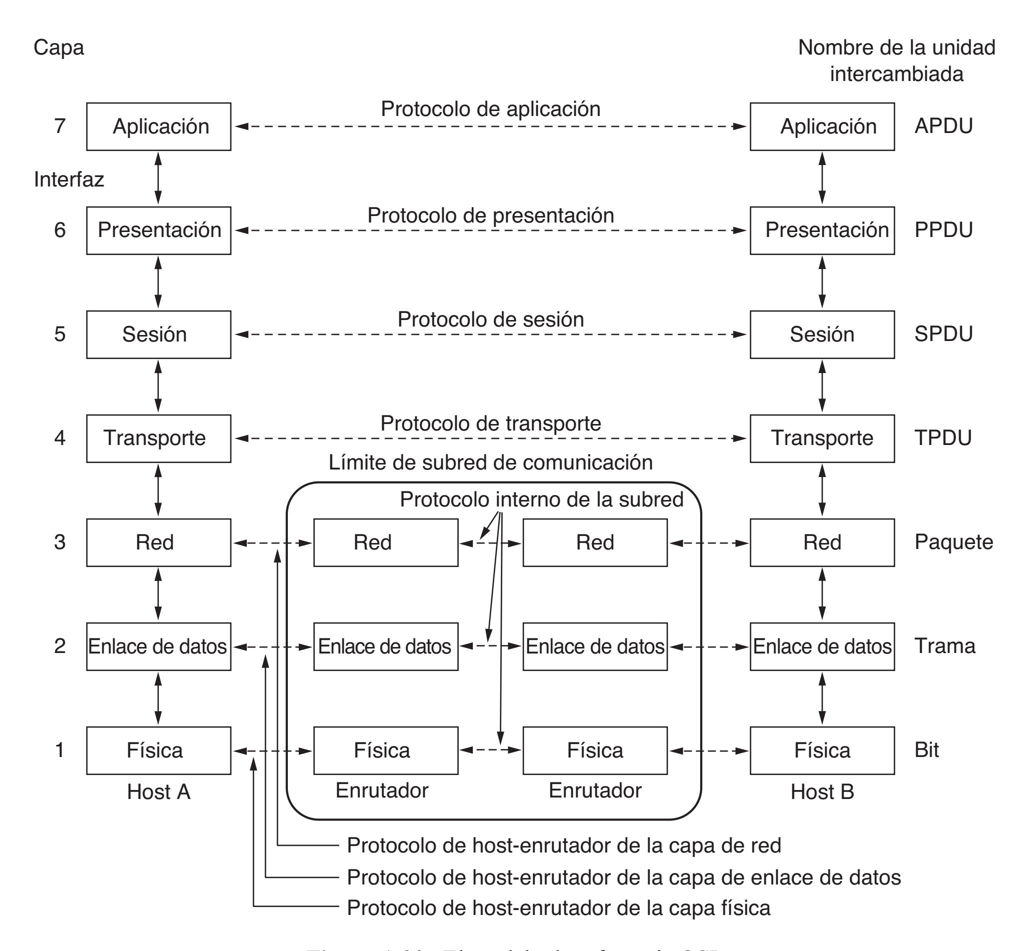


Figura 1-20. El modelo de referencia OSI.

4. Es necesario elegir los límites de las capas de modo que se minimice el flujo de información a través de las interfaces.
5. La cantidad de capas debe ser suficiente como para no tener que agrupar funciones distintas en la misma capa; además, debe ser lo bastante pequeña como para que la arquitectura no se vuelva inmanejable.

A continuación estudiaremos cada capa del modelo en orden, empezando por la capa inferior. Tenga en cuenta que el modelo OSI en sí no es una arquitectura de red, ya que no especifica los servicios y protocolos exactos que se van a utilizar en cada capa. Sólo indica lo que una debe hacer. Sin embargo, la ISO también ha elaborado estándares para todas las capas, aunque no son parte del modelo de referencia en sí. Cada uno se publicó como un estándar internacional separado. Aunque el *modelo* (en parte) es muy usado, los protocolos asociados han estado en el olvido desde hace tiempo.

La capa física

La **capa física** se relaciona con la transmisión de bits puros a través de un canal de transmisión. Los aspectos de diseño tienen que ver con la acción de asegurarse que cuando uno de los lados envíe un bit 1 el otro lado lo reciba como un bit 1, no como un bit 0. En este caso las preguntas típicas son: ¿que señales

eléctricas se deben usar para representar un 1 y un 0?, ¿cuántos nanosegundos dura un bit?, ¿la transmisión puede proceder de manera simultánea en ambas direcciones?, ¿cómo se establece la conexión inicial y cómo se interrumpe cuando ambos lados han terminado?, ¿cuántos pines tiene el conector de red y para qué sirve cada uno? Los aspectos de diseño tienen que ver con las interfaces mecánica, eléctrica y de temporización, así como con el medio de transmisión físico que se encuentra bajo la capa física.

La capa de enlace de datos

La principal tarea de la **capa de enlace de datos** es transformar un medio de transmisión puro en una línea que esté libre de errores de transmisión. Enmascara los errores reales, de manera que la capa de red no los vea. Para lograr esta tarea, el emisor divide los datos de entrada en **tramas de datos** (por lo general, de algunos cientos o miles de bytes) y transmite las tramas en forma secuencial. Si el servicio es confiable, para confirmar la recepción correcta de cada trama, el receptor devuelve una **trama de confirmación de recepción**.

Otra cuestión que surge en la capa de enlace de datos (y en la mayoría de las capas superiores) es cómo evitar que un transmisor rápido inunde de datos a un receptor lento. Tal vez sea necesario algún mecanismo de regulación de tráfico para notificar al transmisor cuando el receptor puede aceptar más datos.

Las redes de difusión tienen una consideración adicional en la capa de enlace de datos: cómo controlar el acceso al canal compartido. Una subcapa especial de la capa de enlace de datos, conocida como subcapa de **control de acceso al medio**, es la que se encarga de este problema.

La capa de red

La **capa de red** controla la operación de la subred. Una cuestión clave de diseño es determinar cómo se encaminan los paquetes desde el origen hasta el destino. Las rutas se pueden basar en tablas estáticas que se "codifican" en la red y rara vez cambian, aunque es más común que se actualicen de manera automática para evitar las fallas en los componentes. También se pueden determinar el inicio de cada conversación; por ejemplo, en una sesión de terminal al iniciar sesión en una máquina remota. Por último, pueden ser muy dinámicas y determinarse de nuevo para cada paquete, de manera que se pueda reflejar la carga actual en la red.

Si hay demasiados paquetes en la subred al mismo tiempo, se interpondrán en el camino unos con otros y formarán cuellos de botella. El manejo de la congestión también es responsabilidad de la capa de red, en conjunto con las capas superiores que adaptan la carga que colocan en la red. Otra cuestión más general de la capa de red es la calidad del servicio proporcionado (retardo, tiempo de tránsito, variaciones, etcétera).

Cuando un paquete tiene que viajar de una red a otra para llegar a su destino, pueden surgir muchos problemas. El direccionamiento utilizado por la segunda red puede ser distinto del que utiliza la primera. La segunda red tal vez no acepte el paquete debido a que es demasiado grande. Los protocolos pueden ser diferentes, etc. Es responsabilidad de la capa de red solucionar todos estos problemas para permitir la interconexión de redes heterogéneas.

En las redes de difusión, el problema de encaminamiento es simple, por lo que con frecuencia la capa de red es delgada o incluso inexistente.

La capa de transporte

La función básica de la **capa de transporte** es aceptar datos de la capa superior, dividirlos en unidades más pequeñas si es necesario, pasar estos datos a la capa de red y asegurar que todas las piezas lleguen

correctamente al otro extremo. Además, todo esto se debe realizar con eficiencia y de una manera que aísle las capas superiores de los inevitables cambios en la tecnología de hardware que se dan con el transcurso del tiempo.

La capa de transporte también determina el tipo de servicio que debe proveer a la capa de sesión y, en última instancia, a los usuarios de la red. El tipo más popular de conexión de transporte es un canal punto a punto libre de errores que entrega los mensajes o bytes en el orden en el que se enviaron. Sin embargo existen otros posibles tipos de servicio de transporte, como el de mensajes aislados sin garantía sobre el orden de la entrega y la difusión de mensajes a múltiples destinos. El tipo de servicio se determina al establecer la conexión (cabe mencionar que es imposible lograr un canal libre de errores; lo que se quiere decir en realidad con este término es que la tasa de errores es lo bastante baja como para ignorarla en la práctica).

La capa de transporte es una verdadera capa de extremo a extremo; lleva los datos por toda la ruta desde el origen hasta el destino. En otras palabras, un programa en la máquina de origen lleva a cabo una conversación con un programa similar en la máquina de destino mediante el uso de los encabezados en los mensajes y los mensajes de control. En las capas inferiores cada uno de los protocolos está entre una máquina y sus vecinos inmediatos, no entre las verdaderas máquinas de origen y de destino, que pueden estar separadas por muchos enrutadores. En la figura 1-20 se muestra la diferencia entre las capas de la 1 a la 3, que están encadenadas, y entre las capas de la 4 a la 7, que son de extremo a extremo.

La capa de sesión

La capa de sesión permite a los usuarios en distintas máquinas establecer **sesiones** entre ellos. Las sesiones ofrecen varios servicios, incluyendo el **control del diálogo** (llevar el control de quién va a transmitir), el **manejo de tokens** (evitar que dos partes intenten la misma operación crítica al mismo tiempo) y la **sincronización** (usar puntos de referencia en las transmisiones extensas para reanudar desde el último punto de referencia en caso de una interrupción).

La capa de presentación

A diferencia de las capas inferiores, que se enfocan principalmente en mover los bits de un lado a otro, la **capa de presentación** se enfoca en la sintaxis y la semántica de la información transmitida. Para hacer posible la comunicación entre computadoras con distintas representaciones internas de datos, podemos definir de una manera abstracta las estructuras de datos que se van a intercambiar, junto con una codificación estándar que se use “en el cable”. La capa de presentación maneja estas estructuras de datos abstractas y permite definir e intercambiar estructuras de datos de mayor nivel (por ejemplo, registros bancarios).

La capa de aplicación

La **capa de aplicación** contiene una variedad de protocolos que los usuarios necesitan con frecuencia. Un protocolo de aplicación muy utilizado es **HTTP** (**Protocolo de Transferencia de Hipertexto**, del inglés *HyperText Transfer Protocol*), el cual forma la base para la World Wide Web. Cuando un navegador desea una página web, envía el nombre de la página que quiere al servidor que la hospeda mediante el uso de HTTP. Después el servidor envía la página de vuelta. Hay otros protocolos de aplicación que se utilizan para transferir archivos, enviar y recibir correo electrónico y noticias.

1.4.2 El modelo de referencia TCP/IP

Pasemos ahora del modelo de referencia OSI al modelo de referencia que se utiliza en la más vieja de todas las redes de computadoras de área amplia: ARPANET y su sucesora, Internet. Aunque más adelante veremos una breve historia de ARPANET, es conveniente mencionar ahora unos cuantos aspectos de esta red. ARPANET era una red de investigación patrocinada por el **DoD** (**Departamento de Defensa de Estados Unidos**, del inglés *U.S. Department of the Defense*). En un momento dado llegó a conectar cientos de universidades e instalaciones gubernamentales mediante el uso de líneas telefónicas rentadas. Cuando después se le unieron las redes de satélites y de radio, los protocolos existentes tuvieron problemas para interactuar con ellas, de modo que se necesitaba una nueva arquitectura de referencia. Así, casi desde el principio la habilidad de conectar varias redes sin problemas fue uno de los principales objetivos de diseño. Posteriormente esta arquitectura se dio a conocer como el **Modelo de referencia TCP/IP**, debido a sus dos protocolos primarios. Este modelo se definió por primera vez en Cerf y Kahn (1974); después se refinó y definió como estándar en la comunidad de Internet (Braden, 1989). Clark (1988) describe la filosofía de diseño detrás de este modelo.

Debido a la preocupación del DoD de que alguno de sus valiosos hosts, enrutadores y puertas de enlace de interredes pudieran ser volados en pedazos en cualquier momento por un ataque de la antigua Unión Soviética, otro de los objetivos principales fue que la red pudiera sobrevivir a la pérdida de hardware de la subred sin que se interrumpieran las conversaciones existentes. En otras palabras, el DoD quería que las conexiones permanecieran intactas mientras las máquinas de origen y de destino estuvieran funcionando, incluso aunque algunas de las máquinas o líneas de transmisión en el trayecto dejaran de funcionar en forma repentina. Además, como se tenían en mente aplicaciones con requerimientos divergentes que abarcaban desde la transferencia de archivos hasta la transmisión de voz en tiempo real, se necesitaba una arquitectura flexible.

La capa de enlace

Todos estos requerimientos condujeron a la elección de una red de conmutación de paquetes basada en una capa sin conexión que opera a través de distintas redes. La capa más baja en este modelo es la **capa de enlace**; ésta describe qué enlaces (como las líneas seriales y Ethernet clásica) se deben llevar a cabo para cumplir con las necesidades de esta capa de interred sin conexión. En realidad no es una capa en el sentido común del término, sino una interfaz entre los hosts y los enlaces de transmisión. El primer material sobre el modelo TCP/IP tiene poco que decir sobre ello.

La capa de interred

Esta capa es el eje que mantiene unida a toda la arquitectura. Aparece en la figura 1-21 con una correspondencia aproximada a la capa de red de OSI. Su trabajo es permitir que los hosts inyecten paquetes en cualquier red y que viajen de manera independiente hacia el destino (que puede estar en una red distinta). Incluso pueden llegar en un orden totalmente diferente al orden en que se enviaron, en cuyo caso es responsabilidad de las capas más altas volver a ordenarlos, si se desea una entrega en orden. Tenga en cuenta que aquí utilizamos "interred" en un sentido genérico, aunque esta capa esté presente en la Internet.

La analogía aquí es con el sistema de correos convencional (lento). Una persona puede dejar una secuencia de cartas internacionales en un buzón en un país y, con un poco de suerte, la mayoría de ellas se entregarán a la dirección correcta en el país de destino. Es probable que las cartas pasen a través de una o más puertas de enlace de correo internacionales en su trayecto, pero esto es transparente a los usuarios. Además, los usuarios no necesitan saber que cada país (es decir, cada red) tiene sus propias estampillas, tamaños de sobre preferidos y reglas de entrega.

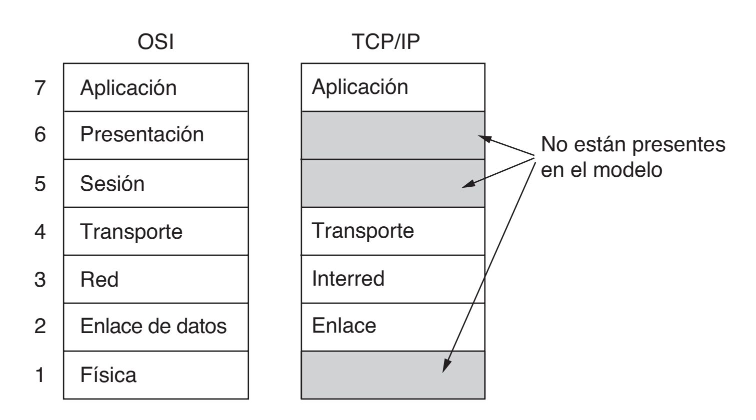


Figura 1-21. El modelo de referencia TCP/IP.

La capa de interred define un formato de paquete y un protocolo oficial llamado **IP** (**Protocolo de Internet**, del inglés *Internet Protocol*), además de un protocolo complementario llamado **ICMP** (**Protocolo de Mensajes de Control de Internet**, del inglés *Internet Control Message Protocol*) que le ayuda a funcionar. La tarea de la capa de interred es entregar los paquetes IP a donde se supone que deben ir. Aquí el ruteo de los paquetes es sin duda el principal aspecto, al igual que la congestión (aunque el IP no ha demostrado ser efectivo para evitar la congestión).

La capa de transporte

Por lo general, a la capa que está arriba de la capa de interred en el modelo TCP/IP se le conoce como **capa de transporte**; y está diseñada para permitir que las entidades pares, en los nodos de origen y de destino, lleven a cabo una conversación, al igual que en la capa de transporte de OSI. Aquí se definieron dos protocolos de transporte de extremo a extremo. El primero, **TCP** (**Protocolo de Control de la Transmisión**, del inglés *Transmission Control Protocol*), es un protocolo confiable orientado a la conexión que permite que un flujo de bytes originado en una máquina se entregue sin errores a cualquier otra máquina en la interred. Este protocolo segmenta el flujo de bytes entrante en mensajes discretos y pasa cada uno a la capa de interred. En el destino, el proceso TCP receptor vuelve a ensamblar los mensajes recibidos para formar el flujo de salida. El TCP también maneja el control de flujo para asegurar que un emisor rápido no pueda inundar a un receptor lento con más mensajes de los que pueda manejar.

El segundo protocolo en esta capa, **UDP** (**Protocolo de Datagrama de Usuario**, del inglés *User Datagram Protocol*), es un protocolo sin conexión, no confiable para aplicaciones que no desean la asignación de secuencia o el control de flujo de TCP y prefieren proveerlos por su cuenta. También se utiliza mucho en las consultas de petición-respuesta de una sola ocasión del tipo cliente-servidor, y en las aplicaciones en las que es más importante una entrega oportuna que una entrega precisa, como en la transmisión de voz o video. En la figura 1-22 se muestra la relación entre IP, TCP y UDP. Desde que se desarrolló el modelo, el IP se ha implementado en muchas otras redes.

La capa de aplicación

El modelo TCP/IP no tiene capas de sesión o de presentación, ya que no se consideraron necesarias. Las aplicaciones simplemente incluyen cualquier función de sesión y de presentación que requieran. La experiencia con el modelo OSI ha demostrado que esta visión fue correcta: estas capas se utilizan muy poco en la mayoría de las aplicaciones.

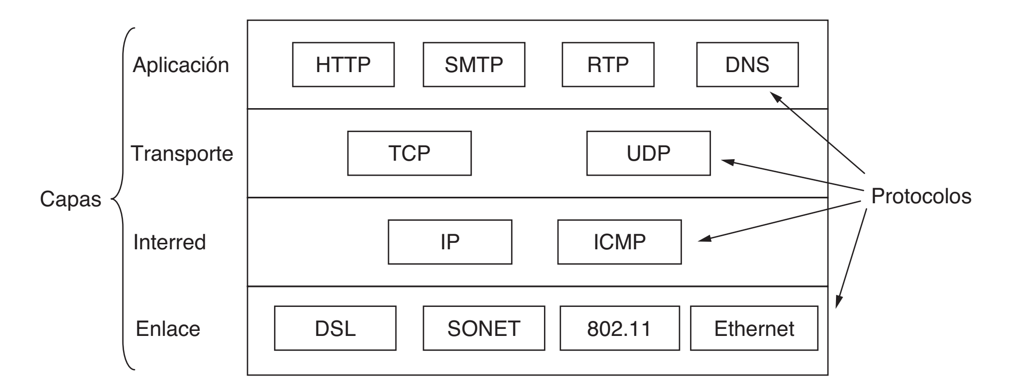


Figura 1-22. El modelo TCP/IP con algunos de los protocolos.

Encima de la capa de transporte se encuentra la **capa de aplicación**. Ésta contiene todos los protocolos de alto nivel. Entre los primeros protocolos están el de terminal virtual (TELNET), transferencia de archivos (FTP) y correo electrónico (SMTP). A través de los años se han agregado muchos otros protocolos. En la figura 1-22 se muestran algunos de los más importantes que veremos más adelante: el Sistema de nombres de dominio (DNS) para resolución de nombres de hosts a sus direcciones de red; HTTP, el protocolo para recuperar páginas de la World Wide Web; y RTP, el protocolo para transmitir medios en tiempo real, como voz o películas.

1.4.3 El modelo utilizado en este libro

Como dijimos antes, la fortaleza del modelo de referencia OSI es el *modelo* en sí (excepto las capas de presentación y de sesión), el cual ha demostrado ser excepcionalmente útil para hablar sobre redes de computadoras. En contraste, la fortaleza del modelo de referencia TCP/IP son los *protocolos*, que se han utilizado mucho durante varios años. Como a los científicos de computadoras les gusta hacer sus propias herramientas, utilizaremos el modelo híbrido de la figura 1-23 como marco de trabajo para este libro.



Figura 1-23. El modelo de referencia que usaremos en este libro.

Este modelo tiene cinco capas, empezando por la capa física, pasando por las capas de enlace, red y transporte hasta llegar a la capa de aplicación. La capa física especifica cómo transmitir bits a través de distintos tipos de medios como señales eléctricas (u otras señales analógicas). La capa de enlace trata sobre cómo enviar mensajes de longitud finita entre computadoras conectadas de manera directa con niveles específicos de confiabilidad. Ethernet y 802.11 son ejemplos de protocolos de capa de enlace.

La capa de red se encarga de combinar varios enlaces múltiples en redes, y redes de redes en interredes, de manera que podamos enviar paquetes entre computadoras distantes. Aquí se incluye la tarea de buscar la ruta por la cual enviarán los paquetes. IP es el principal protocolo de ejemplo que estudiaremos para esta capa. La capa de transporte fortalece las garantías de entrega de la capa de Red, por lo general con una mayor confiabilidad, además provee abstracciones en la entrega, como un flujo de bytes confiable, que coincida con las necesidades de las distintas aplicaciones. TCP es un importante ejemplo de un protocolo de capa de transporte.

Por último, la capa de aplicación contiene programas que hacen uso de la red. Muchas aplicaciones en red tienen interfaces de usuario, como un navegador web. Sin embargo, nuestro interés está en la parte del programa que utiliza la red. En el caso del navegador web se trata del protocolo HTTP. También hay programas de soporte importantes en la capa de aplicación, como el DNS, que muchas aplicaciones utilizan.

La secuencia de nuestros capítulos se basa en este modelo. De esta forma, retenemos el valor del modelo OSI para comprender las arquitecturas de red al tiempo que nos concentramos principalmente en los protocolos que son importantes en la práctica, desde TCP/IP y los protocolos relacionados hasta los más recientes como 802.11, SONET y Bluetooth.

1.4.4 Comparación de los modelos de referencia OSI y TCP/IP

Los modelos de referencia OSI y TCP/IP tienen mucho en común. Ambos se basan en el concepto de una pila de protocolos independientes. Además, la funcionalidad de las capas es muy similar. Por ejemplo, en ambos modelos las capas por encima de la de transporte, incluyendo ésta, se encuentran ahí para proporcionar un servicio de transporte independiente de la red, de extremo a extremo, para los procesos que desean comunicarse. Estas capas forman el proveedor de transporte. También en ambos modelos, las capas que están arriba de la de transporte son usuarias orientadas a la aplicación del servicio de transporte.

A pesar de estas similitudes fundamentales, los dos modelos también tienen muchas diferencias. En esta sección nos enfocaremos en las diferencias clave entre los dos modelos de referencia. Es importante tener en cuenta que aquí compararemos los *modelos de referencia* y no las *pilas de protocolos* correspondientes. Más adelante estudiaremos los protocolos en sí. Un libro completo dedicado a comparar y contrastar TCP/IP y OSI es el de Piscitello y Chapin (1993).

Hay tres conceptos básicos para el modelo OSI:

1. Servicios.
2. Interfaces.
3. Protocolos.

Quizá, la mayor contribución del modelo OSI es que hace explícita la distinción entre estos tres conceptos. Cada capa desempeña ciertos *servicios* para la capa que está sobre ella. La definición del servicio indica lo que hace la capa, no cómo acceden a ella las entidades superiores ni cómo funciona. Define la semántica de la capa.

La *interfaz* de una capa indica a los procesos superiores cómo pueden acceder a ella. Especifica cuáles son los parámetros y qué resultados se pueden esperar. Pero no dice nada sobre su funcionamiento interno.

Por último, la capa es la que debe decidir qué *protocolos* de iguales utilizar. Puede usar los protocolos que quiera, siempre y cuando realice el trabajo (es decir, que provea los servicios ofrecidos). También los puede cambiar a voluntad sin afectar el software de las capas superiores.

Estas ideas encajan muy bien con las ideas modernas sobre la programación orientada a objetos. Al igual que una capa, un objeto tiene un conjunto de métodos (operaciones) que los procesos fuera

del objeto pueden invocar. La semántica de estos métodos define el conjunto de servicios que ofrece el objeto. Los parámetros y resultados de los métodos forman la interfaz del objeto. El código interno del objeto es su protocolo y no se puede ver ni es de la incumbencia de las entidades externas al objeto.

Al principio, el modelo TCP/IP no tenía una distinción clara entre los servicios, las interfaces y los protocolos, aunque las personas han tratado de reajustarlo a fin de hacerlo más parecido al OSI. Por ejemplo, los únicos servicios que realmente ofrece la capa de interred son SEND IP PACKET y RECEIVE IP PACKET. Como consecuencia, los protocolos en el modelo OSI están ocultos de una mejor forma que en el modelo TCP/IP, además se pueden reemplazar con relativa facilidad a medida que la tecnología cambia. La capacidad de realizar dichos cambios con transparencia es uno de los principales propósitos de tener protocolos en capas en primer lugar.

El modelo de referencia OSI se ideó *antes* de que se inventaran los protocolos correspondientes. Este orden significa que el modelo no estaba orientado hacia un conjunto específico de protocolos, un hecho que lo hizo bastante general. La desventaja de este orden fue que los diseñadores no tenían mucha experiencia con el tema y no supieron bien qué funcionalidad debían colocar en cada una de las capas.

Por ejemplo, en un principio la capa de enlace de datos trabajaba sólo con redes de punto a punto. Cuando surgieron las redes de difusión, fue necesario insertar una nueva subcapa al modelo. Además, cuando las personas empezaron a construir redes reales mediante el modelo OSI y los protocolos existentes, se descubrió que estas redes no coincidían con las especificaciones de los servicios requeridos, de modo que tuvieron que integrar en el modelo subcapas convergentes que permitieran cubrir las diferencias. Finalmente, el comité en un principio esperaba que cada país tuviera una red operada por el gobierno en la que se utilizaran los protocolos OSI, por lo que no se tomó en cuenta la interconexión de redes. Para no hacer el cuento largo, las cosas no salieron como se esperaba.

Con TCP/IP sucedió lo contrario: primero llegaron los protocolos y el modelo era en realidad sólo una descripción de los protocolos existentes. No hubo problema para que los protocolos se ajustaran al modelo. Encajaron a la perfección. El único problema fue que el *modelo* no encajaba en ninguna otra pila de protocolos. En consecuencia, no era útil para describir otras redes que no fueran TCP/IP.

Pasando de las cuestiones filosóficas a las más específicas, una diferencia obvia entre los dos modelos está en el número de capas: el modelo OSI tiene siete capas, mientras que el modelo TCP/IP tiene cuatro. Ambos tienen capas de (inter)red, transporte y aplicación, pero las demás capas son distintas.

Hay otra diferencia en el área de la comunicación sin conexión frente a la comunicación orientada a conexión. El modelo OSI soporta ambos tipos de comunicación en la capa de red, pero sólo la comunicación orientada a conexión en la capa de transporte, en donde es más importante (ya que el servicio de transporte es visible a los usuarios). El modelo TCP/IP sólo soporta un modo en la capa de red (sin conexión) pero soporta ambos en la capa de transporte, de manera que los usuarios tienen una alternativa, que es muy importante para los protocolos simples de petición-respuesta.

1.4.5 Una crítica al modelo y los protocolos OSI

Ni el modelo OSI y sus protocolos, ni el modelo TCP/IP y sus protocolos son perfectos. Ambos pueden recibir bastantes críticas, y así se ha hecho. En ésta y en la siguiente sección analizaremos algunas de ellas. Empezaremos con el modelo OSI y después examinaremos el modelo TCP/IP.

Para cuando se publicó la segunda edición de este libro (1989), a muchos expertos en el campo les pareció que el modelo OSI y sus protocolos iban a adueñarse del mundo y sacar todo lo demás a su paso.

Pero esto no fue así. ¿Por qué? Tal vez sea útil analizar en retrospectiva algunas de las razones. Podemos resumirlas de la siguiente manera:

1. Mala sincronización.
2. Mala tecnología.
3. Malas implementaciones.
4. Mala política.

Mala sincronización

Veamos la razón número uno: mala sincronización. El tiempo en el cual se establece un estándar es absolutamente imprescindible para su éxito. David Clark, del Massachusetts Institute of Technology (MIT), tiene una teoría de estándares a la que llama el *apocalipsis de los dos elefantes*, la cual se ilustra en la figura 1-24.

Esta figura muestra la cantidad de actividad alrededor de un nuevo tema. Cuando se descubre el tema por primera vez, hay una ráfaga de actividades de investigación en forma de discusiones, artículos y reuniones. Después de cierto tiempo esta actividad disminuye, las corporaciones descubren el tema y llega la ola de inversión de miles de millones de dólares.

Es imprescindible que los estándares se escriban en el intermedio entre los dos "elefantes". Si se escriben demasiado pronto (antes de que los resultados de la investigación estén bien establecidos), tal vez el tema no se entienda bien todavía; el resultado es un estándar malo. Si se escriben demasiado tarde, es probable que muchas empresas hayan hecho ya importantes inversiones en distintas maneras de hacer las cosas, de modo que los estándares se ignorarán en la práctica. Si el intervalo entre los dos elefantes es muy corto (ya que todos tienen prisa por empezar), la gente que desarrolla los estándares podría quedar aplastada.

En la actualidad, parece que los protocolos estándar de OSI quedaron aplastados. Para cuando aparecieron los protocolos de OSI, los protocolos TCP/IP competidores ya se utilizaban mucho en universidades que hacían investigaciones. Aunque todavía no llegaba la ola de inversión de miles de millones de dólares, el mercado académico era lo bastante grande como para que muchos distribuidores empezaran a ofrecer con cautela los productos TCP/IP. Para cuando llegó el modelo OSI, los distribuidores no quisieron apoyar una segunda pila de protocolos hasta que se vieron obligados a hacerlo, de modo que no hubo ofertas iniciales. Como cada empresa estaba esperando a que otra tomara la iniciativa, ninguna lo hizo y OSI nunca se llevó a cabo.

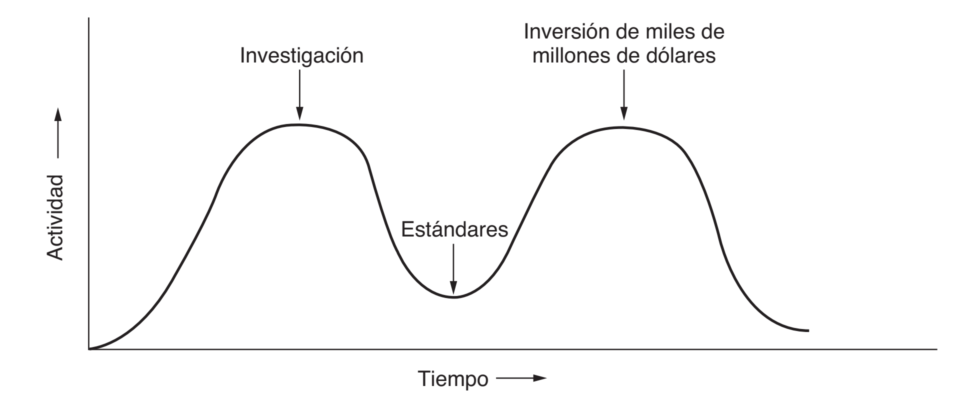


Figura 1-24. El apocalipsis de los dos elefantes.

Mala tecnología

La segunda razón por la que OSI nunca tuvo éxito fue que tanto el modelo como los protocolos tienen fallas. La opción de siete capas era más política que técnica, además de que dos de las capas (sesión y presentación) están casi vacías, mientras que otras dos (enlace de datos y red) están demasiado llenas.

El modelo OSI, junto con sus correspondientes definiciones y protocolos de servicios, es muy complejo. Si se apilan, los estándares impresos ocupan una fracción considerable de un metro de papel. Además son difíciles de implementar e ineficientes en su operación. En este contexto nos viene a la mente un acertijo propuesto por Paul Mockapetris y citado por Rose (1993):

P: ¿Qué obtenemos al cruzar un pandillero con un estándar internacional?
R: Alguien que le hará una oferta que no podrá comprender.

Además de ser incomprensible, otro problema con el modelo OSI es que algunas funciones como el direccionamiento, el control de flujo y el control de errores, vuelven a aparecer una y otra vez en cada capa. Por ejemplo, Saltzer y sus colaboradores (1984) han señalado que para ser efectivo, hay que llevar a cabo el control de errores en la capa más alta, por lo que repetirlo una y otra vez en cada una de las capas más bajas es con frecuencia innecesario e ineficiente.

Malas implementaciones

Dada la enorme complejidad del modelo y los protocolos, no es sorprendente que las implementaciones iniciales fueran enormes, pesadas y lentas. Todos los que las probaron se arrepintieron. No tuvo que pasar mucho tiempo para que las personas asociaran "OSI" con la "mala calidad". Aunque los productos mejoraron con el tiempo, la imagen perduró.

En contraste, una de las primeras implementaciones de TCP/IP fue parte del UNIX, de Berkeley, y era bastante buena (y además, gratuita). Las personas empezaron a utilizarla rápidamente, lo cual provocó que se formara una extensa comunidad de usuarios, lo que condujo a mejoras, lo que llevó a una comunidad todavía mayor. En este caso la espiral fue hacia arriba, en vez de ir hacia abajo.

Malas políticas

Gracias a la implementación inicial, mucha gente (en especial los académicos) pensaba que TCP/IP era parte de UNIX, y UNIX en la década de 1980 para los académicos era algo así como la paternidad (que en ese entonces se consideraba erróneamente como maternidad) y el pay de manzana para los estadounidenses comunes.

Por otro lado, OSI se consideraba en muchas partes como la invención de los ministerios europeos de telecomunicaciones, de la Comunidad Europea y después, del gobierno de Estados Unidos. Esta creencia no era del todo justificada, pero la simple idea de un grupo de burócratas gubernamentales que trataban de obligar a los pobres investigadores y programadores que estaban en las trincheras desarrollando verdaderas redes de computadoras a que adoptaran un estándar técnicamente inferior no fue de mucha utilidad para la causa de OSI. Algunas personas vieron este suceso como algo similar a cuando IBM anunció en la década de 1960 que PL/I era el lenguaje del futuro, o cuando luego el DoD corrigió esto para anunciar que en realidad el lenguaje era Ada.

1.4.6 Una crítica al modelo de referencia TCP/IP

El modelo y los protocolos de TCP/IP también tienen sus problemas. Primero, el modelo no diferencia con claridad los conceptos de servicios, interfaces y protocolos. La buena práctica de la ingeniería

de software requiere una distinción entre la especificación y la implementación, algo que OSI hace con mucho cuidado y que TCP/IP no. En consecuencia, el modelo TCP/IP no sirve mucho de guía para diseñar modernas redes que utilicen nuevas tecnologías.

Segundo, el modelo TCP/IP no es nada general y no es muy apropiado para describir cualquier pila de protocolos aparte de TCP/IP. Por ejemplo, es imposible tratar de usar el modelo TCP/IP para describir Bluetooth.

Tercero, la capa de enlace en realidad no es una capa en el sentido normal del término como se utiliza en el contexto de los protocolos en capas. Es una interfaz (entre las capas de red y de enlace de datos). La diferencia entre una interfaz y una capa es crucial, y hay que tener mucho cuidado al respecto.

Cuarto, el modelo TCP/IP no distingue entre la capa física y la de enlace de datos. Éstas son completamente distintas. La capa física trata sobre las características de transmisión del cable de cobre, la fibra óptica y la comunicación inalámbrica. La tarea de la capa de enlace de datos es delimitar el inicio y el fin de las tramas, además de transmitirlas de un extremo al otro con el grado deseado de confiabilidad. Un modelo apropiado debe incluir ambas capas por separado. El modelo TCP/IP no hace esto.

Por último, aunque los protocolos IP y TCP se diseñaron e implementaron con sumo cuidado, muchos de los otros protocolos se fueron creando según las necesidades del momento, producidos generalmente por un par de estudiantes de licenciatura que los mejoraban hasta fastidiarse. Después las implementaciones de los protocolos se distribuían en forma gratuita, lo cual trajo como consecuencia que se utilizaran amplia y profundamente en muchas partes y, por ende, eran difíciles de reemplazar. Algunos de ellos son un poco vergonzosos en la actualidad. Por ejemplo, el protocolo de terminal virtual TELNET se diseñó para una terminal de Teletipo mecánica de 10 caracteres por segundo. No sabe nada sobre las interfaces gráficas de usuario y los ratones. Sin embargo, aún se sigue usando a 30 años de su creación.

1.5 REDES DE EJEMPLO

El tema de las redes de computadoras cubre muchos tipos distintos de redes, grandes y pequeñas, populares y no tanto. Tienen distintos objetivos, escalas y tecnologías. En las siguientes secciones analizaremos algunos ejemplos para tener una idea de la variedad que podemos encontrar en el área de las redes de computadoras.

Empezaremos con Internet, que tal vez sea la red más popular; analizaremos su historia, evolución y tecnología. Después consideraremos la red de teléfonos móviles. Técnicamente es muy distinta de Internet y contrasta muy bien con ella. Más adelante introduciremos el IEEE 802.11, el estándar dominante para las redes LAN inalámbricas. Por último, analizaremos las redes RFID y de sensores, tecnologías que extienden el alcance de la red para incluir al mundo físico y los objetos cotidianos.

1.5.1 Internet

En realidad Internet no es una red, sino una enorme colección de distintas redes que utilizan ciertos protocolos comunes y proveen ciertos servicios comunes. Es un sistema inusual en cuanto a que nadie la planeó y nadie la controla. Para comprender mejor esto, empecemos desde el inicio para ver cómo se ha desarrollado y por qué. Si desea leer una maravillosa historia de Internet, le recomendamos ampliamente el libro de Jim Naughton (2000). Es uno de esos libros inusuales que no sólo son divertidos, sino que también cuenta con 20 páginas de *ibídems* y *obras citadas* (*ob. cit.*) para el verdadero historiador. Una parte del material de esta sección se basa en ese libro.

Claro que también se han escrito innumerables libros sobre Internet y sus protocolos. Para obtener más información puede consultar a Maufer (1999).

ARPANET

La historia empieza a finales de la década de 1950. En la cúspide de la Guerra Fría, el DoD de Estados Unidos quería una red de comando y control que pudiera sobrevivir a una guerra nuclear. En ese tiempo todas las comunicaciones militares utilizaban la red telefónica pública, que se consideraba vulnerable. Podemos ver la razón de esta creencia en la figura 1-25(a). Los puntos negros representan las oficinas de conmutación telefónica, cada una de las cuales se conectaba a miles de teléfonos. Estas oficinas de conmutación se conectaban a su vez con oficinas de conmutación de mayor nivel (oficinas interurbanas), para formar una jerarquía nacional con sólo una pequeña cantidad de redundancia. La vulnerabilidad del sistema era que, si se destruían unas cuantas oficinas interurbanas clave, se podía fragmentar el sistema en muchas islas aisladas.

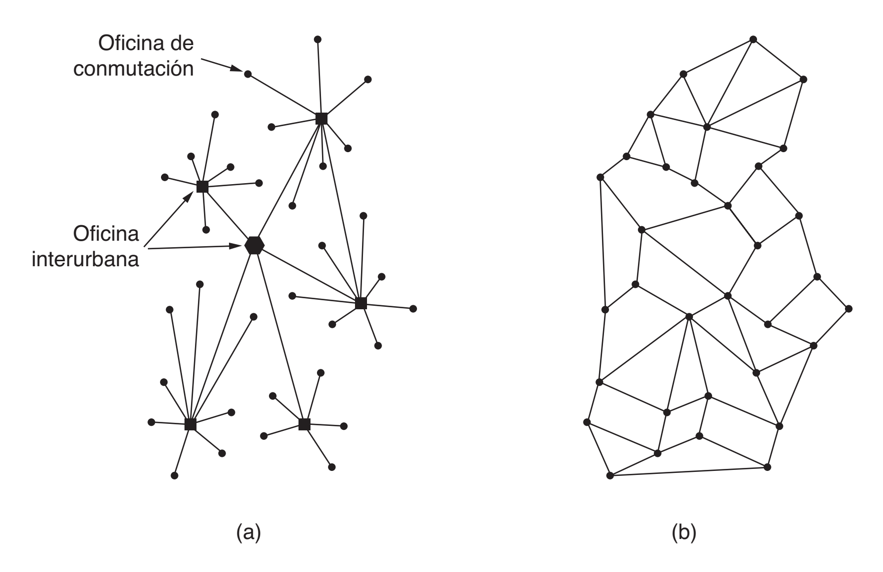


Figura 1-25. (a) Estructura de un sistema telefónico. (b) El sistema de conmutación distribuida propuesto por Baran.

Alrededor de la década de 1960, el DoD otorgó un contrato a la empresa RAND Corporation para buscar una solución. Uno de sus empleados, Paul Baran, ideó el diseño tolerante a fallas altamente distribuido de la figura 1-25(b). Como las rutas entre dos oficinas de conmutación cualesquiera eran ahora mucho más largas de lo que las señales análogas podían viajar sin distorsión, Baran propuso el uso de la tecnología de conmutación de paquetes digital, y escribió varios informes para el DoD en donde describió sus ideas con detalle (Baran, 1964). A los oficiales del Pentágono les gustó el concepto y pidieron a AT&T, que en ese entonces era el monopolio telefónico nacional en Estados Unidos, que construyera un prototipo. Pero AT&T hizo caso omiso de las ideas de Baran. La corporación más grande y opulenta del mundo no iba a permitir que un joven impertinente les dijera cómo construir un sistema telefónico. Dijeron que la idea de Baran no se podía construir y se desechó.

Pasaron otros siete años y el DoD seguía sin poder obtener un mejor sistema de comando y control. Para comprender lo que ocurrió después tenemos que remontarnos hasta octubre de 1957, cuando la antigua Unión Soviética venció a Estados Unidos en la carrera espacial con el lanzamiento del primer satélite artificial, Sputnik. Cuando el presidente Eisenhower trató de averiguar quién se había quedado dormido en los controles, quedó consternado al descubrir que el Ejército, la Marina y la Fuerza Aérea estaban riñendo por el presupuesto de investigación del Pentágono. Su respuesta inmediata fue crear una sola

organización de investigación de defensa, **ARPA** (**Agencia de Proyectos de Investigación Avanzados**, del inglés *Advanced Research Projects Agency*). La ARPA no tenía científicos ni laboratorios; de hecho, sólo tenía una oficina y un pequeño presupuesto (según los estándares del Pentágono). Para realizar su trabajo otorgaba concesiones y contratos a las universidades y las compañías cuyas ideas fueran prometedoras.

Durante los primeros años, la ARPA trató de averiguar cuál debería ser su misión. En 1967 Larry Roberts, director de la ARPA, quien trataba de averiguar cómo proveer acceso remoto a las computadoras, giró su atención a las redes. Contactó a varios expertos para decidir qué hacer. Uno de ellos de nombre Wesley Clark, sugirió construir una subred de conmutación de paquetes y conectar cada host a su propio enrutador.

Después de cierto escepticismo inicial, Roberts aceptó la idea y presentó un documento algo impreciso sobre ella en el Simposio SIGOPS de la ACM sobre Principios de Sistemas Operativos que se llevó a cabo en Gatlinburg, Tennessee, a finales de 1967 (Roberts, 1967). Para gran sorpresa de Roberts había otro documento en la conferencia que describía un sistema similar que no sólo se había diseñado, sino que también se había implementado por completo bajo la dirección de Donald Davies en el Laboratorio Nacional de Física (NPL), en Inglaterra. El sistema del NPL no era un sistema nacional (sólo conectaba varias computadoras en su campus), pero demostraba que la conmutación de paquetes podía funcionar. Además citaba el trabajo anterior de Baran que había sido descartado. Roberts regresó de Gatlinburg determinado a construir lo que después se convirtió en **ARPANET**.

La subred consistiría de minicomputadoras llamadas **IMP** (**Procesadores de Mensajes de Interfaz**, del inglés *Interface Message Processors*), conectadas por líneas de transmisión de 56 kbps. Para una confiabilidad alta, cada IMP se conectaría por lo menos a otras dos. La subred sería de datagramas, de manera que si se destruían algunas líneas e IMP, los mensajes se podrían encaminar nuevamente de manera automática a través de rutas alternativas.

Cada nodo de la red debía estar constituido por una IMP y un host, en el mismo cuarto, conectados por un cable corto. Un host podía enviar mensajes de hasta 8 063 bits a su IMP, que a su vez los descompondría en paquetes de 1 008 bits a lo más y los enviaría de manera independiente a su destino. Cada paquete se recibía en su totalidad antes de enviarlo, por lo que la subred fue la primera red electrónica de conmutación de paquetes de almacenamiento y envío.

Entonces ARPA lanzó una convocatoria para construir la subred y fueron 12 compañías las que licitaron. Después de evaluar todas las propuestas, la ARPA seleccionó a BBN, una empresa de consultoría con base en Cambridge, Massachusetts, y en diciembre de 1968 le otorgó un contrato para construir la subred y escribir el software. BBN optó por usar como IMP las minicomputadoras Honeywell DDP-316 modificadas de manera especial con palabras de 16 bits y 12 KB de memoria básica. Los IMP no tenían discos, ya que las partes móviles se consideraban no confiables. Los IMP se interconectaron mediante líneas de 56 kbps que se rentaban a las compañías telefónicas. Aunque ahora 56 kbps son la opción para los adolescentes que no pueden pagar DSL o cable, en ese entonces era lo mejor que el dinero podía comprar.

El software se dividió en dos partes: subred y host. El software de subred consistía del extremo IMP de la conexión host a IMP, del protocolo IMP a IMP y de un protocolo de IMP de origen a IMP de destino diseñado para mejorar la confiabilidad. En la figura 1-26 se muestra el diseño original de la ARPANET.

Fuera de la subred también se necesitaba software, es decir, el extremo host de la conexión host a IMP, el protocolo host a host y el software de aplicación. Pronto quedó claro que BBN consideraba que al aceptar un mensaje en un cable host a IMP y colocarlo en el cable host a IMP de destino, su trabajo estaba terminado.

Pero Roberts tenía un problema: los hosts también necesitaban software. Para lidiar con ello, convocó una junta de investigadores de redes, que en su mayor parte eran estudiantes de licenciatura, en Snowbird, Utah, en el verano de 1969. Los estudiantes esperaban que un experto en redes les explicara el gran diseño de la red y su software, y que después les asignara la tarea de escribir parte de ella. Quedaron pasmados

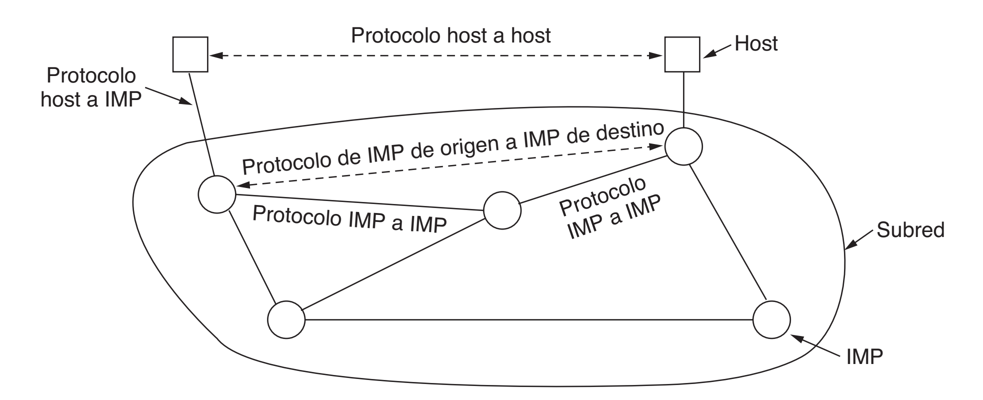


Figura 1-26. Diseño original de ARPANET.

al descubrir que no había ningún experto en redes ni un gran diseño. Tuvieron que averiguar qué hacer por su cuenta.

Sin embargo, de alguna forma una red experimental se puso en línea en diciembre de 1969 con cuatro nodos: en UCLA, UCSB, SRI y la Universidad de Utah. Se eligieron estos cuatro nodos debido a que todos tenían una gran cantidad de contratos de ARPA y todos tenían computadoras host distintas y totalmente incompatibles (sólo para hacerlo más divertido). Dos meses antes se había enviado el primer mensaje de host a host desde el nodo de UCLA por un equipo dirigido por Len Kleinrock (pionero de la teoría de conmutación de paquetes), hasta el nodo de SRI. La red creció con rapidez a medida que se entregaban e instalaban más equipos IMP; pronto abarcó Estados Unidos. En la figura 1-27 se muestra qué tan rápido creció ARPANET durante los primeros tres años.

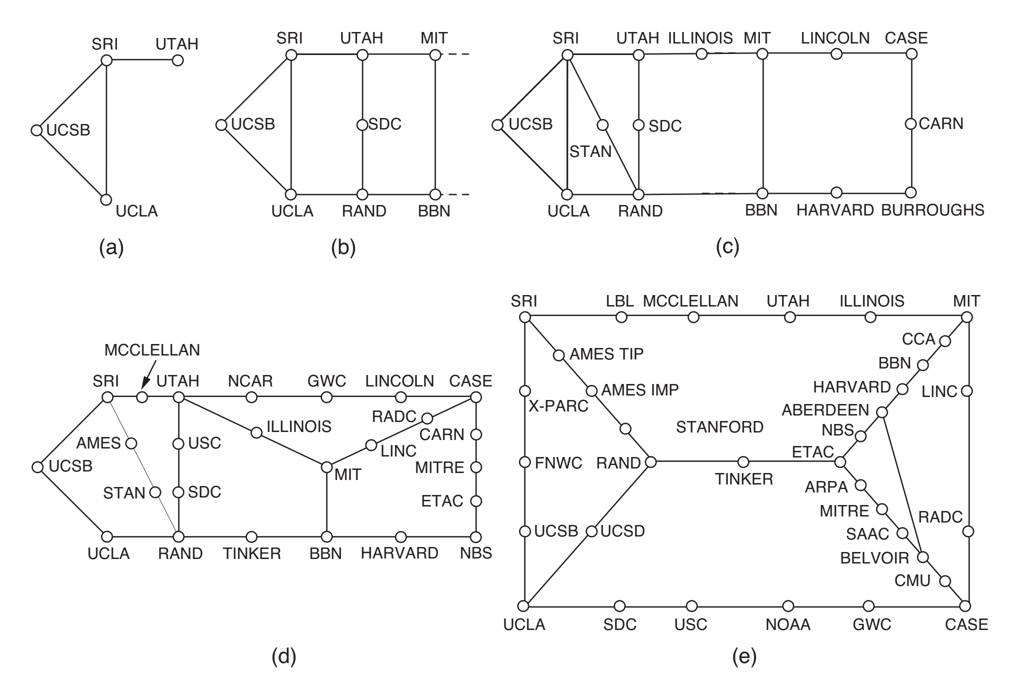


Figura 1-27. Crecimiento de ARPANET. (a) Diciembre de 1969. (b) Julio de 1970. (c) Marzo de 1971. (d) Abril de 1972. (e) Septiembre de 1972.

Además de ayudar al crecimiento de la recién creada ARPANET, la ARPA también patrocinó la investigación sobre el uso de las redes satelitales y las redes de radio de paquetes móviles. En una famosa demostración, un camión que recorría California usó la red de radio de paquetes para enviar mensajes a SRI, que a su vez los envió a través de ARPANET a la Costa Este, en donde se enviaron al Colegio Universitario, en Londres, a través de la red satelital. Gracias a esto, un investigador en el camión pudo utilizar una computadora en Londres mientras conducía por California.

Este experimento también demostró que los protocolos existentes de ARPANET no eran adecuados para trabajar en distintas redes. Esta observación condujo a más investigaciones sobre protocolos, lo que culminó con la invención del modelo y los protocolos TCP/IP (Cerf y Kahn, 1974). El modelo TCP/IP se diseñó de manera específica para manejar la comunicación a través de interredes, algo que se volvía día con día más importante a medida que más redes se conectaban a ARPANET.

Para fomentar la adopción de estos nuevos protocolos, la ARPA otorgó varios contratos para implementar TCP/IP en distintas plataformas de computadora, incluyendo sistemas de IBM, DEC y HP, así como para el UNIX, de Berkeley. Los investigadores de la Universidad de California, en Berkeley, rediseñaron el modelo TCP/IP con una nueva interfaz de programación llamada **sockets** para la futura versión 4.2BSD del UNIX, de Berkeley. También escribieron muchos programas de aplicación, utilería y administración para mostrar lo conveniente que era usar la red con sockets.

La sincronización era perfecta. Muchas universidades acababan de adquirir una segunda o tercera computadoras VAX y una LAN para conectarlas, pero no tenían software de red. Cuando llegó el 4.2BSD junto con TCP/IP, los sockets y muchas utilerías de red, el paquete completo se adoptó de inmediato. Además, con TCP/IP era fácil conectar las redes LAN a ARPANET, y muchas lo hicieron.

Durante la década de 1980 se conectaron redes adicionales (en especial redes LAN) a ARPANET. A medida que aumentó la escala, el proceso de buscar hosts se hizo cada vez más costoso, por lo que se creó el **DNS** (**Sistema de Nombres de Dominio**, del inglés *Domain Name System*) para organizar a las máquinas en dominios y resolver nombres de host en direcciones IP. Desde entonces, el DNS se convirtió en un sistema de base de datos distribuido y generalizado para almacenar una variedad de información relacionada con la asignación de nombres. En el capítulo 7 estudiaremos este sistema con detalle.

NSFNET

A finales de la década de 1970, la **NSF** (**Fundación Nacional de la Ciencia**, del inglés U.S. National Science Foundation) vio el enorme impacto que había tenido ARPANET en la investigación universitaria al permitir que científicos de todo el país compartieran datos y colaboraran en proyectos de investigación. Pero para entrar a ARPANET una universidad tenía que tener un contrato de investigación con el DoD. Como muchas no tenían un contrato, la respuesta inicial de la NSF fue patrocinar la Red de Ciencias Computacionales (**CSNET**, del inglés *Computer Science Network*) en 1981. Esta red conectó los departamentos de ciencias computacionales y los laboratorios de investigación industrial a ARPANET por medio de líneas de marcación y rentadas. A finales de la década de 1980, la NSF fue más allá y decidió diseñar un sucesor para ARPANET que estuviera abierto a todos los grupos universitarios de investigación.

Para tener algo concreto con qué empezar, la NSF decidió construir una red troncal (*backbone*) para conectar sus seis centros de supercomputadoras en San Diego, Boulder, Champaign, Pittsburgh, Ithaca y Princeton. Cada supercomputadora recibió un hermano pequeño que consistía en una microcomputadora LSI-11 llamada **fuzzball**. Las fuzzballs se conectaron a líneas rentadas de 56 kbps para formar la subred, la misma tecnología de hardware que utilizaba ARPANET. Sin embargo, la tecnología de software era diferente: las fuzzballs funcionaban con TCP/IP desde un principio, así que se convirtió en la primera WAN de TCP/IP.

La NSF también patrocinó algunas redes regionales (finalmente fueron cerca de 20) que se conectaban a la red troncal para permitir que los usuarios de miles de universidades, laboratorios de investigación,

bibliotecas y museos tuvieran acceso a cualquiera de las supercomputadoras y se comunicaran entre sí. La red completa, incluyendo la red troncal y las redes regionales, se llamó **NSFNET**. Se conectaba a ARPANET por medio de un enlace entre un IMP y una fuzzball en el cuarto de máquinas de Carnegie-Mellon. En la figura 1-28 se ilustra la primera red troncal de NSFNET, superpuesta en un mapa de Estados Unidos.

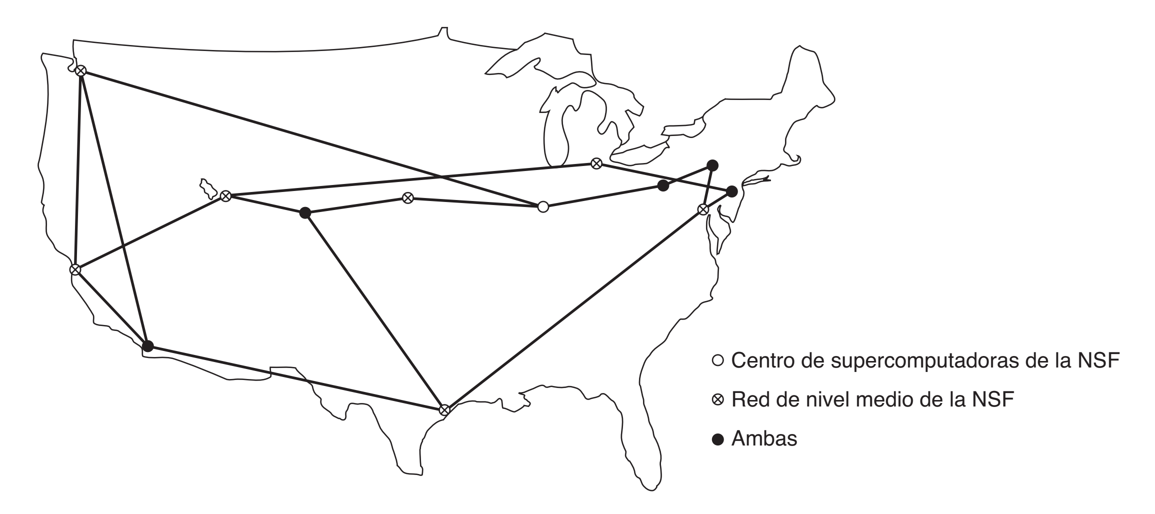


Figura 1-28. La red troncal de NSFNET en 1988.

La NSFNET fue un éxito instantáneo y se sobrecargó desde el principio. La NSF empezó de inmediato a planear su sucesora y otorgó un contrato al consorcio MERIT con base en Michigan para llevar a cabo la tarea. Se rentaron a MCI (que desde entonces se fusionó con WorldCom) unos canales de fibra óptica a 448 kbps para proveer la versión 2 de la red troncal. Se utilizaron equipos PC-RT de IBM como enrutadores. Esta red también se sobrecargó casi de inmediato y, para 1990, la segunda red troncal se actualizó a 1.5 Mbps.

Mientras la red seguía creciendo, la NSF se dio cuenta de que el gobierno no podría seguir financiando el uso de las redes por siempre. Además, las organizaciones comerciales querían unirse pero los estatutos de la NSF les prohibían usar las redes pagadas por la Fundación. En consecuencia, la NSF animó a MERIT, MCI e IBM para que formaran una corporación sin fines de lucro llamada **ANS** (**Redes y Servicios Avanzados**, del inglés *Advanced Networks and Services*), como primer paso en el camino hacia la comercialización. En 1990, ANS se hizo cargo de la NSFNET y actualizó los enlaces de 1.5 Mbps a 45 Mbps para formar la **ANSNET**. Esta red operó durante cinco años y después se vendió a America Online. Pero para entonces, varias empresas estaban ofreciendo el servicio IP comercial y era evidente que el gobierno debía ahora salirse del negocio de las redes.

Para facilitar la transición y asegurarse de que cada red regional se pudiera comunicar con las demás redes regionales, la NSF otorgó contratos a cuatro distintos operadores de red para establecer un **NAP** (**Punto de Acceso a la Red**, del inglés *Network Access Point*). Estos operadores fueron PacBell (San Francisco), Ameritech (Chicago), MFS (Washington, D.C.) y Sprint (Nueva York, en donde para fines de NAP, Pennsauken, Nueva Jersey cuenta como la ciudad de Nueva York). Todos los operadores de redes que quisieran ofrecer el servicio de red troncal a las redes regionales de la NSF se tenían que conectar a todos los NAP.

Este arreglo significaba que un paquete que se originara en cualquier red regional podía elegir entre varias portadoras de red troncal para ir desde su NAP hasta el NAP de destino. En consecuencia, las

portadoras de red troncal se vieron forzadas a competir por el negocio de las redes regionales con base en el servicio y al precio, que desde luego era lo que se pretendía. Como resultado, el concepto de una sola red troncal predeterminada se reemplazó por una infraestructura competitiva impulsada por el comercio. A muchas personas les gusta criticar al gobierno federal por no ser innovador, pero en el área de las redes fueron el DoD y la NSF quienes crearon la infraestructura que formó la base para Internet y después la entregaron a la industria para que la pusiera en funcionamiento.

Durante la década de 1990, muchos otros países y regiones también construyeron redes de investigación nacional, que con frecuencia seguían el patrón de ARPANET y de la NSFNET. Entre éstas tenemos a EuropaNET y EBONE en Europa, que empezaron con líneas de 2 Mbps y después actualizaron a líneas de 34 Mbps. En un momento dado, la infraestructura de red en Europa también se puso en manos de la industria.

Internet ha cambiado mucho desde sus primeros días. Su tamaño se expandió de manera considerable con el surgimiento de la World Wide Web (WWW) a principios de la década de 1990. Datos recientes de Internet Systems Consortium indican que el número de hosts visibles en Internet está cerca de los 600 millones. Ésta es una estimación baja, pero excede por mucho los varios millones de hosts que había cuando se sostuvo la primera conferencia sobre la WWW en el CERN en 1994.

También ha cambiado mucho la forma en que usamos Internet. Al principio dominaban las aplicaciones como el correo electrónico para los académicos, los grupos de noticias, inicios remotos de sesión y transferencias de archivos. Después cambió a correo para todos, luego la web y la distribución de contenido de igual a igual, como el servicio Napster que está cerrado en la actualidad. Ahora están empezando a tomar popularidad la distribución de medios en tiempo real, las redes sociales (como Facebook) y los microblogs (como Twitter). Estos cambios trajeron a Internet tipos de medios más complejos, y por ende, mucho más tráfico. De hecho, el tráfico dominante en Internet parece cambiar con cierta regularidad puesto que, por ejemplo, las nuevas y mejores formas de trabajar con la música o las películas se pueden volver muy populares con gran rapidez.

Arquitectura de Internet

La arquitectura de Internet también cambió mucho debido a que creció en forma explosiva. En esta sección trataremos de analizar de manera breve las generalidades sobre cómo se ve Internet en la actualidad. La imagen se complica debido a las continuas fusiones en los negocios de las compañías telefónicas (telcos), las compañías de cable y los ISP, y por lo que es difícil distinguir quién hace cada cosa. Uno de los impulsores de esta confusión es la convergencia de las telecomunicaciones, en donde una red se utiliza para distintos servicios que antes realizaban distintas compañías. Por ejemplo, en un "triple play", una compañía le puede vender telefonía, TV y servicio de Internet a través de la misma conexión de red, con el supuesto de que usted ahorrará dinero. En consecuencia, la descripción aquí proporcionada será algo más simple que la realidad. Y lo que es verdad hoy tal vez no lo sea mañana.

En la figura 1-29 se muestra el panorama completo. Examinaremos esta figura pieza por pieza, empezando con una computadora en el hogar (en los extremos de la figura). Para unirse a Internet, la computadora se conecta a un **Proveedor de servicios de Internet**, o simplemente **ISP**, a quien el usuario compra **acceso** o **conectividad a Internet**. Esto permite a la computadora intercambiar paquetes con todos los demás hosts accesibles en Internet. El usuario podría enviar paquetes para navegar por la web o para cualquiera de los otros miles de usos, en realidad no importa. Hay muchos tipos de acceso a Internet y por lo general se distinguen con base en el ancho de banda que se ofrece además de su costo, pero el atributo más importante es la conectividad.

Una manera común de conectar un ISP es mediante la línea telefónica, en cuyo caso su compañía telefónica será su ISP. La tecnología **DSL** (**Línea de Suscriptor Digital**, del inglés *Digital Subscriber Line*) reutiliza la línea telefónica que se conecta a su casa para obtener una transmisión de datos digital. La

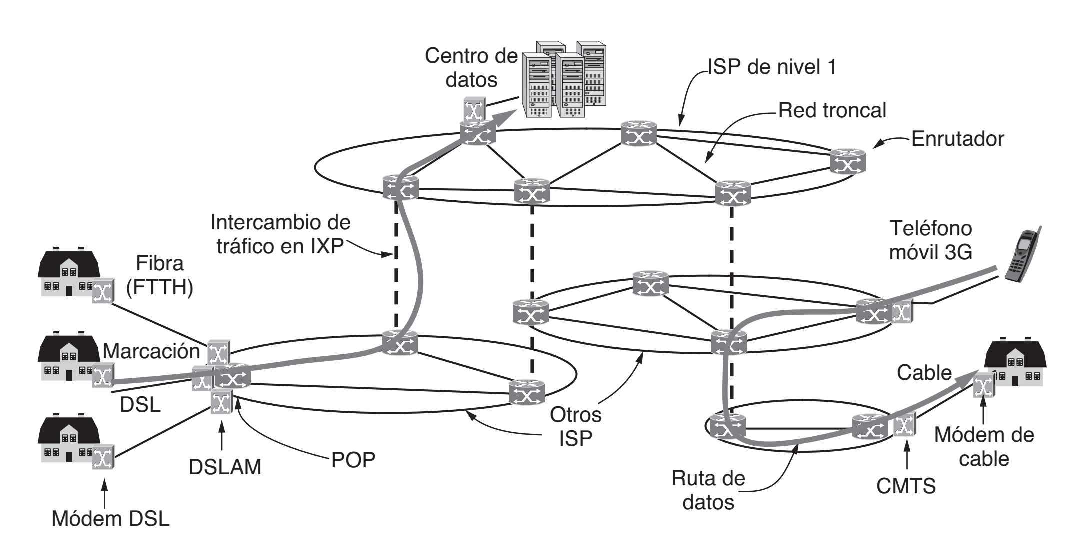


Figura 1-29. Generalidades sobre la arquitectura de Internet.

computadora se conecta a un dispositivo conocido como **módem DSL**, el cual realiza la conversión entre los paquetes digitales y las señales analógicas que pueden pasar libremente a través de la línea telefónica. En el otro extremo hay un dispositivo llamado **DSLAM** (**Multiplexor de Acceso a la Línea de Suscriptor Digital**, del inglés *Digital Subscriber Line Access Multiplexer*) que realiza la conversión entre señales y paquetes.

Hay otras formas populares de conectarse a un ISP, las cuales se muestran en la figura 1-29. DSL es una opción de utilizar la línea telefónica local con más ancho de banda que la acción de enviar bits a través de una llamada telefónica tradicional en vez de una conversación de voz. A esto último se le conoce como **marcación** y se lleva a cabo con un tipo distinto de módem en ambos extremos. La palabra **módem** es la abreviación de "*mo*dulador *dem*odulador" y se refiere a cualquier dispositivo que realiza conversiones entre bits digitales y señales analógicas.

Otro método es enviar señales a través del sistema de TV por cable. Al igual que DSL, ésta es una forma de reutilizar la infraestructura existente, que en este caso es a través de los canales de TV por cable que no se utilizan. El dispositivo en el extremo conectado a la casa se llama **módem de cable** y el dispositivo en la **cabecera del cable** se llama **CMTS** (**Sistema de Terminación del Módem de Cable**, del inglés *Cable Modem Termination System*).

Las tecnologías DSL y de TV por cable proveen acceso a Internet con velocidades que varían desde una pequeña fracción de un megabit/segundo hasta varios megabits/segundo, dependiendo del sistema. Estas velocidades son mucho mayores que en las líneas de marcación, las cuales se limitan a 56 kbps debido al estrecho ancho de banda que se utiliza para las llamadas de voz. Al acceso a Internet con una velocidad mucho mayor que la de marcación se le llama **banda ancha**. El nombre hace referencia al ancho de banda más amplio que se utiliza para redes más veloces, en vez de hacer referencia a una velocidad específica.

Los métodos de acceso mencionados hasta ahora se limitan con base en el ancho de banda de la "última milla" o último tramo de transmisión. Al usar cable de fibra óptica en las residencias, se puede proveer un acceso más rápido a Internet con velocidades en el orden de 10 a 100 Mbps. A este diseño se le conoce como **FTTH** (**Fibra para el Hogar**, del inglés *Fiber To The Home*). Para los negocios en áreas comerciales tal vez tenga sentido rentar una línea de transmisión de alta velocidad de las oficinas hasta el ISP más cercano. Por ejemplo, en Estados Unidos una línea T3 opera aproximadamente a 45 Mbps.

La tecnología inalámbrica también se utiliza para acceder a Internet. Un ejemplo que veremos en breve es el de las redes de teléfonos móviles 3G. Estas redes pueden proveer una transmisión de datos a velocidades de 1 Mbps o mayores para los teléfonos móviles y los suscriptores fijos que se encuentren en el área de cobertura.

Ahora podemos mover los paquetes entre el hogar y el ISP. A la ubicación en la que los paquetes entran a la red del ISP para que se les dé servicio le llamamos el **POP** (**Punto De Presencia**, del inglés *Point Of Presence*) del ISP. A continuación explicaremos cómo se mueven los paquetes entre los POP de distintos ISP. De aquí en adelante, el sistema es totalmente digital y utiliza la conmutación de paquetes.

Las redes de ISP pueden ser de alcance regional, nacional o internacional. Ya hemos visto que su arquitectura está compuesta de líneas de transmisión de larga distancia que interconectan enrutadores en los POP de las distintas ciudades a las que los ISP dan servicio. A este equipo se le denomina la **red troncal** (*backbone*) del ISP. Si un paquete está destinado a un host al que el ISP da servicio directo, ese paquete se encamina a través de la red troncal y se entrega al host. En caso contrario, se debe entregar a otro ISP.

Los ISP conectan sus redes para intercambiar tráfico en lo que llamamos un **IXP** (**Punto de Intercambio en Internet**, del inglés *Internet eXchange Points*). Se dice que los ISP conectados **intercambian tráfico** entre sí. Hay muchos IXP en ciudades de todo el mundo. Se dibujan en sentido vertical en la figura 1-29 debido a que las redes de ISP se traslapan geográficamente. En esencia, un IXP es un cuarto lleno de enrutadores, por lo menos uno por ISP. Una LAN en el cuarto conecta a todos los enrutadores, de modo que los paquetes se pueden reenviar desde cualquier red troncal de ISP a cualquier otra red troncal de ISP. Los IXP pueden ser instalaciones extensas pertenecientes a entidades independientes. Uno de los más grandes es Amsterdam Internet Exchange, en donde se conectan cientos de ISP y a través del cual intercambian cientos de gigabits/segundo de tráfico.

El intercambio de tráfico (*peering*) que ocurre en los IXP depende de las relaciones comerciales entre los ISP. Hay muchas relaciones posibles. Por ejemplo, un ISP pequeño podría pagar a un ISP más grande para obtener conectividad a Internet para alcanzar hosts distantes, así como cuando un cliente compra servicio a un proveedor de Internet. En este caso, se dice que el ISP pequeño paga por el **tránsito**. O tal vez dos ISP grandes decidan intercambiar tráfico de manera que cada ISP pueda entregar cierto tráfico al otro ISP sin tener que pagar por el tránsito. Una de las diversas paradojas de Internet es que los ISP que compiten públicamente por los clientes, cooperan con frecuencia en forma privada para intercambiar tráfico (Metz, 2001).

La ruta que toma un paquete por Internet depende de las opciones de intercambio de tráfico de los ISP. Si el ISP que va a entregar un paquete intercambia tráfico con el ISP de destino, podría entregar el paquete directamente a su igual. En caso contrario, podría encaminar el paquete hasta el lugar más cercano en donde se conecte con un proveedor de tránsito pagado, de manera que éste pueda entregar el paquete. En la figura 1-29 se dibujan dos rutas de ejemplo a través de los ISP. Es muy común que la ruta que toma un paquete no sea la ruta más corta a través de Internet.

En la parte superior de la “cadena alimenticia” se encuentra un pequeño grupo de empresas, como AT&T y Sprint, que operan extensas redes troncales internacionales con miles de enrutadores conectados mediante enlaces de fibra óptica con un extenso ancho de banda. Estos ISP no pagan por el tránsito. Por lo general se les denomina ISP de **nivel 1** y se dice que forman la red troncal de Internet, ya que todos los demás se tienen que conectar a ellos para poder llegar a toda la Internet.

Las empresas que proveen mucho contenido, como Google y Yahoo!, tienen sus computadoras en **centros de datos** que están bien conectados al resto de Internet. Estos centros de datos están diseñados para computadoras, no para humanos, y pueden contener estante (*rack*) tras estante de máquinas, a lo que llamamos **granja de servidores**. Los centros de datos de **colocación** u **hospedaje** permiten a los clientes tener equipo como servidores en los POP de un ISP, de manera que se puedan realizar conexiones cortas y rápidas entre los servidores y las redes troncales del ISP. La industria de hospedaje en Internet se

está virtualizando cada vez más, de modo que ahora es común rentar una máquina virtual que se ejecuta en una granja de servidores en vez de instalar una computadora física. Estos centros de datos son tan grandes (decenas o cientos de miles de máquinas) que la electricidad es uno de los principales costos, por lo que algunas veces estos centros de datos se construyen en áreas en donde el costo de la electricidad sea más económico.

Con esto terminamos nuestra breve introducción a Internet. En los siguientes capítulos tendremos mucho qué decir sobre los componentes individuales y su diseño, los algoritmos y los protocolos. Algo más que vale la pena mencionar aquí es que el significado de estar en Internet está cambiando. Antes se decía que una máquina estaba en Internet si: (1) ejecutaba la pila de protocolos TCP/IP; (2) tenía una dirección IP; y (3) podía enviar paquetes IP a todas las demás máquinas en Internet. Sin embargo, a menudo los ISP reutilizan las direcciones dependiendo de las computadoras que se estén utilizando en un momento dado, y es común que las redes domésticas compartan una dirección IP entre varias computadoras. Esta práctica quebranta la segunda condición. Las medidas de seguridad, como los firewalls, también pueden bloquear en parte las computadoras para que no reciban paquetes, con lo cual se quebranta la tercera condición. A pesar de estas dificultades, tiene sentido decir que esas máquinas estarán en Internet mientras permanezcan conectadas a sus ISP.

También vale la pena mencionar que algunas compañías han interconectado todas sus redes internas existentes, y con frecuencia usan la misma tecnología que Internet. Por lo general, se puede acceder a estas **intranets** sólo desde las premisas de la compañía o desde computadoras notebook de la empresa, pero en los demás aspectos funcionan de la misma manera que Internet.

1.5.2 Redes de teléfonos móviles de tercera generación

A las personas les encanta hablar por teléfono mucho más de lo que les gusta navegar en Internet, y esto ha logrado que la red de teléfonos móviles sea la más exitosa del mundo. Tiene más de cuatro mil millones de suscriptores a nivel mundial. Para poner esta cantidad en perspectiva, digamos que constituye aproximadamente 60% de la población mundial y es mucho más que la cantidad de hosts de Internet y líneas telefónicas fijas combinadas (ITU, 2009).

La arquitectura de la red de teléfonos móviles ha cambiado y ha crecido de manera considerable durante los últimos 40 años. Los sistemas de telefonía móvil de primera generación transmitían las llamadas de voz como señales de variación continua (analógicas) en vez de secuencias de bits (digitales). El sistema **AMPS** (**Sistema Telefónico Móvil Avanzado**, del inglés *Advanced Mobile Phone System*), que se desarrolló en Estados Unidos en 1982, fue un sistema de primera generación muy popular. Los sistemas de teléfonos móviles de segunda generación cambiaron a la transmisión de las llamadas de voz en formato digital para aumentar su capacidad, mejorar la seguridad y ofrecer mensajería de texto. El sistema **GSM** (**Sistema Global para Comunicaciones Móviles**, del inglés *Global System for Mobile communications*), que se implementó a partir de 1991 y se convirtió en el sistema de telefonía móvil más utilizado en el mundo, es un sistema 2G.

Los sistemas de tercera generación (o 3G) comenzaron a implementarse en el año 2001 y ofrecen servicios de datos tanto de voz digital como de datos digitales de banda ancha. También vienen con mucho lenguaje tecnológico y distintos estándares a elegir. La ITU (una organización internacional de estándares de la que hablaremos en la siguiente sección) define al estándar 3G en sentido general como un servicio que ofrece velocidades de por lo menos 2 Mbps para usuarios estacionarios o móviles, y de 384 kbps en un vehículo en movimiento. El sistema **UMTS** (**Sistema Universal de Telecomunicaciones Móviles**, del inglés *Universal Mobile Telecommunications System*), también conocido como **WCDMA** (**Acceso Múltiple por División de Código de Banda Ancha**, del inglés *Wideband Code Division Multiple Access*), es el principal sistema 3G que se está implementando con rapidez en todo el mundo. Puede proveer hasta

14 Mbps en el enlace de bajada y casi 6 Mbps en el enlace de subida. Las futuras versiones utilizarán varias antenas y radios para proveer velocidades aún mayores para los usuarios.

El recurso escaso en los sistemas 3G, al igual que en los sistemas 2G y 1G anteriores, es el espectro de radio. Los gobiernos conceden el derecho de usar partes del espectro a los operadores de la red de telefonía móvil, a menudo mediante una subasta de espectro en donde los operadores de red realizan ofertas. Es más fácil diseñar y operar sistemas cuando se tiene una parte del espectro con licencia, ya que a nadie más se le permite transmitir en ese espectro, pero la mayoría de las veces es algo muy costoso. Por ejemplo, en el Reino Unido en el año 2000, se subastaron cinco licencias para 3G por un total aproximado de $40 mil millones de dólares.

Esta escasez del espectro es la que condujo al diseño de la **red celular** que se muestra en la figura 1-30 y que ahora se utiliza en las redes de telefonía móvil. Para manejar la interferencia de radio entre los usuarios, el área de cobertura se divide en celdas. Dentro de una celda, a los usuarios se les asignan canales que no interfieren entre sí y que no provocan mucha interferencia para las celdas adyacentes. Esto permite una reutilización eficiente del espectro, o **reutilización de frecuencia**, en las celdas adyacentes, lo cual incrementa la capacidad de la red. En los sistemas 1G, que transmitían cada llamada de voz en una banda de frecuencia específica, las frecuencias se elegían con cuidado de modo que no tuvieran conflictos con las celdas adyacentes. De esta forma, una frecuencia dada sólo se podría reutilizar una vez en varias celdas. Los sistemas 3G modernos permiten que cada celda utilice todas las frecuencias, pero de una manera que resulte en un nivel tolerable de interferencia para las celdas adyacentes. Existen variaciones en el diseño celular, incluyendo el uso de antenas direccionales o sectorizadas en torres de celdas para reducir aún más la interferencia, pero la idea básica es la misma.

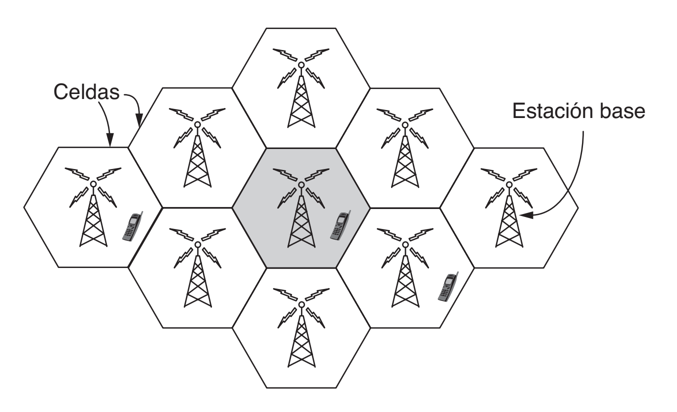


Figura 1-30. Diseño celular de las redes de telefonía móvil.

La arquitectura de la red de telefonía móvil es muy distinta a la de Internet. Tiene varias partes, como se muestra en la versión simplificada de la arquitectura UMTS en la figura 1-31. Primero tenemos a la **interfaz aérea**. Éste es un término elegante para el protocolo de radiocomunicación que se utiliza a través del aire entre el dispositivo móvil (como el teléfono celular) y la **estación base celular**. Los avances en la interfaz aérea durante las últimas décadas han aumentado en forma considerable las velocidades de datos inalámbricas. La interfaz aérea de UTMS se basa en el **Acceso Múltiple por División de Código** (**CDMA**, del inglés *Code Division Multiple Access*), una técnica que estudiaremos en el capítulo 2.

La estación base celular forma junto con su controlador la **red de acceso por radio**. Esta parte constituye el lado inalámbrico de la red de telefonía móvil. El nodo controlador o **RNC** (**Controlador de la**

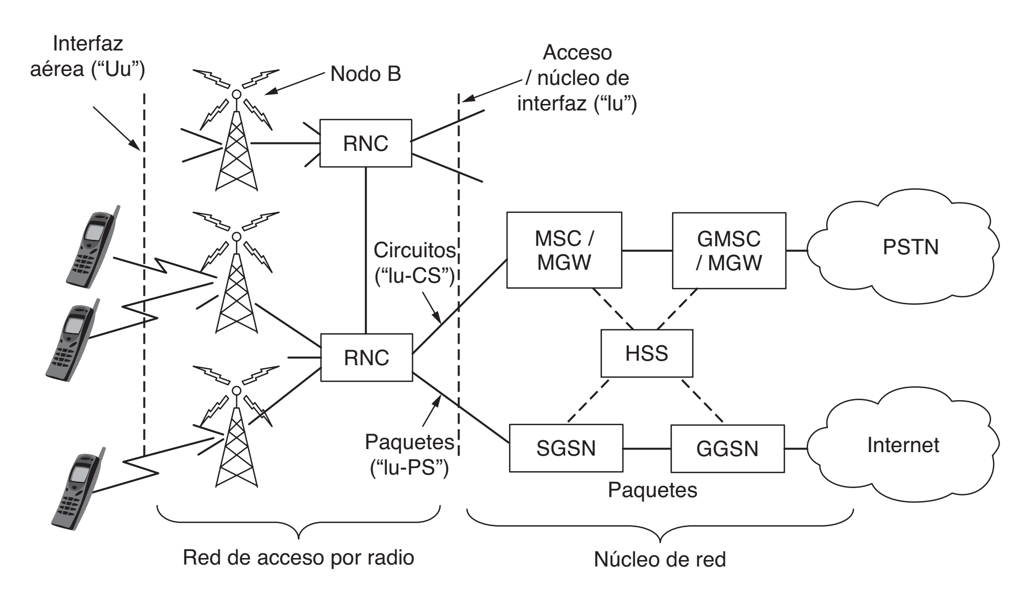


Figura 1-31. Arquitectura de la red de telefonía móvil 3G UTMS.

Red de Radio, del inglés *Radio Network Controller*) controla la forma en que se utiliza el espectro. La estación base implementa a la interfaz aérea. A ésta se le conoce como **Nodo B**, una etiqueta temporal que se quedó para siempre.

El resto de la red de telefonía móvil transporta el tráfico para la red de acceso por radio. A esto se le conoce como **núcleo de red**. La red básica UMTS evolucionó a partir de la red básica que se utilizaba para el sistema GSM 2G anterior. Sin embargo, algo sorprendente está ocurriendo en la red básica UMTS.

Desde los inicios de las redes se ha venido desatando una guerra entre las personas que apoyan las redes de paquetes (es decir, subredes sin conexión) y las personas que apoyan las redes de circuitos (es decir, redes orientadas a conexión). Los principales defensores de los paquetes provienen de la comunidad de Internet. En un diseño sin conexión, cada paquete se encamina de manera independiente a los demás paquetes. Como consecuencia, si algunos enrutadores fallan durante una sesión, no habrá daño alguno siempre y cuando el sistema pueda reconfigurarse a sí mismo en forma dinámica, de modo que los siguientes paquetes puedan encontrar una ruta a su destino, aun cuando sea distinta a la que hayan utilizado los paquetes anteriores.

El campo de circuitos proviene del mundo de las compañías telefónicas. En el sistema telefónico, un usuario debe marcar el número de la parte a la que va a llamar y esperar una conexión antes de poder hablar o enviar datos. Esta forma de realizar la conexión establece una ruta a través del sistema telefónico que se mantiene hasta terminar la llamada. Todas las palabras o paquetes siguen la misma ruta. Si falla una línea o un interruptor en la ruta se aborta la llamada, es decir, es un método menos tolerante a las fallas en comparación con el diseño sin conexión.

La ventaja de los circuitos es que soportan la calidad del servicio con más facilidad. Al establecer una conexión por adelantado, la subred puede reservar recursos como el ancho de banda del enlace, el espacio de búfer de los switches, y tiempo de la CPU. Si alguien intenta hacer una llamada y no hay suficientes recursos disponibles, la llamada se rechaza y el usuario recibe una señal de ocupado. De esta forma, una vez establecida la conexión, recibirá un buen servicio.

Con una red sin conexión, si llegan demasiados paquetes al mismo enrutador en el mismo momento, es probable que pierda algunos. El emisor se dará cuenta de esto en un momento dado y volverá a enviarlos, pero la calidad del servicio será intermitente e inadecuada para transmitir audio o video, a menos que

la red tenga una carga ligera. Sin necesidad de decirlo, proveer una calidad adecuada de audio y video es algo por lo que las compañías telefónicas se preocupan mucho, de aquí que prefieran un servicio orientado a la conexión.

La sorpresa en la figura 1-31 es que hay equipo tanto de paquetes como de conmutación de circuitos en el núcleo de red. Esto muestra a la red de telefonía móvil en transición, en donde las compañías de telefonía móvil pueden implementar una o, en ocasiones, ambas alternativas. Las redes de telefonía móvil antiguas usaban un núcleo de conmutación de paquetes al estilo de la red telefónica tradicional para transmitir las llamadas de voz. Esta herencia se puede ver en la red UMTS con los elementos **MSC** (**Centro de Conmutación Móvil**, del inglés *Mobile Switching Center*), **GMSC** (**Centro de Conmutación Móvil de Puerta de Enlace**, del inglés *Gateway Mobile Switching Center*) y **MGW** (**Puerta de Enlace de Medios**, del inglés *Media Gateway*) que establecen conexiones a través de un núcleo de red con conmutación de paquetes como **PSTN** (**Red Telefónica Pública Conmutada**, del inglés *Public Switched Telephone Network*).

Los servicios de datos se han convertido en una parte de la red de telefonía móvil mucho más importante de lo que solían ser, empezando con la mensajería de texto y los primeros servicios de datos de paquetes, como **GPRS** (**Servicio General de Paquetes de Radio**, del inglés *General Packet Radio Service*) en el sistema GSM. Estos servicios de datos antiguos operaban a decenas de kbps, pero los usuarios querían más. En comparación, una llamada de voz se transmite a una velocidad de 64 kbps, comúnmente de 3 a 4 veces menos con compresión.

Para transmitir todos estos datos, los nodos del núcleo de red UMTS se conectan directamente a una red de conmutación de paquetes. El **SGSN** (**Nodo de Soporte del Servicio GPRS**, del inglés *Serving GPRS Support Node*) y el **GGSN** (**Nodo de Soporte de la Puerta de Enlace de GPRS***, del inglés Gateway GPRS Support Node*) transmiten paquetes de datos hacia y desde dispositivos móviles y hacen interfaz con redes de paquetes externas, como Internet.

Esta transición está destinada a continuar en las redes de telefonía móvil que se planean e implementan en la actualidad. Incluso se utilizan protocolos de Internet en dispositivos móviles para establecer conexiones para llamadas de voz a través de una red de paquetes de datos, en forma de voz sobre IP. El protocolo IP y los paquetes se utilizan en todo el camino, desde el acceso por radio hasta el núcleo de red. Desde luego que también se están haciendo cambios en el diseño de las redes IP para soportar una mejor calidad de servicio. Si no fuera así, los problemas con la señal entrecortada de audio y video no impresionarían a los clientes y dejarían de pagar. En el capítulo 5 retomaremos este tema.

Otra diferencia entre las redes de telefonía móvil y la Internet tradicional es la movilidad. Cuando un usuario se sale del rango de una estación base celular y entra al rango de otra, el flujo de datos se debe encaminar nuevamente desde la estación antigua hasta la nueva estación base celular. A esta técnica se le conoce como **traspaso** (*handover*) o **entrega** (*handoff*) y se ilustra en la figura 1-32.

El dispositivo móvil o la estación base pueden solicitar un traspaso si disminuye la calidad de la señal. En algunas redes celulares (por lo general las que están basadas en tecnología CDMA) es posible conec-

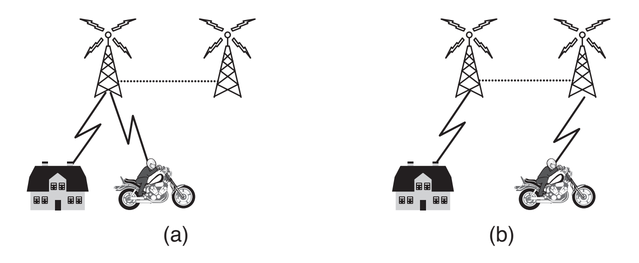


Figura 1-32. Traspaso de telefonía móvil (a) antes, (b) después.

tarse a la nueva estación base antes de desconectarse de la estación anterior. Esto mejora la calidad de la conexión para el dispositivo móvil, ya que no se interrumpe el servicio; el dispositivo móvil se conecta a dos estaciones base por un breve instante. A esta manera de realizar un traspaso se le llama **traspaso suave** para diferenciarla de un **traspaso duro**, en donde el dispositivo móvil se desconecta de la estación base anterior antes de conectarse a la nueva estación.

Una cuestión relacionada es cómo buscar un móvil en primer lugar cuando hay una llamada entrante. Cada red de telefonía móvil tiene un **HSS** (**Servidor de Suscriptores Locales**, del inglés *Home Subscriber Server*) en el núcleo de red, el cual conoce la ubicación de cada suscriptor así como demás información de perfil que se utiliza para la autenticación y la autorización. De esta forma, para encontrar un dispositivo móvil hay que ponerse en contacto con el HSS.

El último tema en cuestión es la seguridad. A través de la historia, las compañías telefónicas han tomado la seguridad mucho más en serio que las compañías de Internet por mucho tiempo, debido a la necesidad de cobrar por el servicio y evitar el fraude (en los pagos). Por desgracia, esto no dice mucho. Sin embargo, en la evolución de la tecnología 1G a la 3G, las compañías de telefonía móvil han sido capaces de desarrollar varios mecanismos básicos de seguridad para dispositivos móviles.

A partir del sistema GSM 2G, el teléfono móvil se dividió en una terminal y un chip removible que contenía la identidad del suscriptor y la información de su cuenta. Al chip se le conoce de manera informal como **tarjeta SIM** (**Módulo de Identidad del Suscriptor**, del inglés *Subscriber Identity Module*). Las tarjetas SIM se pueden usar en distintas terminales para activarlas, además de que proveen una seguridad básica. Cuando los clientes de GSM viajan a otros países por motivos de negocios o de placer, a menudo traen consigo sus terminales pero compran una nueva tarjeta SIM por unos cuantos dólares al llegar, para poder hacer llamadas locales sin cargos de roaming.

Para reducir los fraudes, la red de telefonía móvil también usa la información en las tarjetas SIM para autenticar a los suscriptores y verificar que puedan usar la red. Con el sistema UTMS, el dispositivo móvil también usa la información en la tarjeta SIM para verificar que está hablando con una red legítima.

La privacidad es otro aspecto de la seguridad. Las señales inalámbricas se difunden a todos los receptores cercanos, por lo que para evitar que alguien pueda espiar las conversaciones se utilizan claves criptográficas en la tarjeta SIM para cifrar las transmisiones. Esta metodología ofrece una mayor privacidad que en los sistemas 1G, que se podían intervenir fácilmente, pero no es una panacea debido a las debilidades en los esquemas de cifrado.

Las redes de telefonía móvil están destinadas a desempeñar un papel central en las futuras redes. Ahora tratan más sobre aplicaciones móviles de banda ancha que sobre llamadas de voz, y esto tiene implicaciones importantes para las interfaces aéreas, la arquitectura del núcleo de red y la seguridad de las futuras redes. Las tecnologías 4G que son más veloces y mejores ya están en fase de diseño bajo el nombre de **LTE** (**Evolución a Largo Plazo**, del inglés *Long Term Evolution*), incluso a medida que continúa el diseño y el desarrollo de la tecnología 3G. Hay otras tecnologías inalámbricas que también ofrecen acceso a Internet de banda ancha para clientes fijos y móviles, en particular las redes 802.16 bajo el nombre común de **WiMAX**. Es totalmente posible que LTE y WiMAX vayan a chocar en un futuro y es difícil predecir qué les ocurrirá.

1.5.3 Redes LAN inalámbricas: 802.11

Casi al mismo tiempo en que aparecieron las computadoras laptop, muchas personas soñaban con entrar a una oficina y que su laptop se conectara mágicamente a Internet. En consecuencia, varios grupos empezaron a trabajar en formas para lograr este objetivo. La metodología más práctica consiste en equipar tanto a la oficina como las computadoras laptop con transmisores de radio de corto alcance y receptores para que se puedan comunicar.

El trabajo en este campo condujo rápidamente a que varias empresas empezaran con la comercialización de las redes LAN inalámbricas. El problema era que ni siquiera había dos de ellas que fueran compatibles. La proliferación de estándares implicaba que una computadora equipada con un radio marca *X* no trabajaría en un cuarto equipado con una estación base marca *Y*. A mediados de la década de 1990, la industria decidió que sería muy conveniente tener un estándar para las redes LAN inalámbricas, de modo que el comité IEEE que había estandarizado las redes LAN alámbricas recibió la tarea de idear un estándar para redes LAN inalámbricas.

La primera decisión fue la más sencilla: cómo llamar a este estándar. Todos los demás estándares de LAN tenían números como 802.1, 802.2 y 802.3 hasta 802.10, así que al estándar de LAN inalámbrica se le dio el número 802.11. En la jerga computacional a este estándar se le conoce con el nombre de **WiFi**, pero es un estándar importante y merece respeto, de modo que lo llamaremos por su nombre: 802.11.

El resto fue más difícil. El primer problema era hallar una banda de frecuencia adecuada que estuviera disponible, de preferencia a nivel mundial. La metodología utilizada fue contraria a la que se utilizó en las redes de telefonía móvil. En vez de un espectro costoso bajo licencia, los sistemas 802.11 operan en bandas sin licencia como las bandas **ISM** (**Industriales, Científicas y Médicas**, del inglés *Industrial, Scientific, and Medical*) definidas por el ITU-R (por ejemplo, 902-929 MHz, 2.4-2.5 GHz, 5.725-5.825 GHz). Todos los dispositivos pueden usar este espectro siempre y cuando limiten su potencia de transmisión para dejar que coexistan distintos dispositivos. Desde luego que esto significa que los radios 802.11 podrían entrar en competencia con los teléfonos inalámbricos, los abridores de puertas de garaje y los hornos de microondas.

Las redes 802.11 están compuestas de clientes (como laptops y teléfonos móviles) y de una infraestructura llamada **AP** (**Puntos de Acceso**) que se instala en los edificios. Algunas veces a los puntos de acceso se les llama **estaciones base**. Los puntos de acceso se conectan a la red alámbrica y toda la comunicación entre los clientes se lleva a cabo a través de un punto de acceso. También es posible que los clientes que están dentro del rango del radio se comuniquen en forma directa, como en el caso de dos computadoras en una oficina sin un punto de acceso. A este arreglo se le conoce como **red *ad hoc***. Se utiliza con menor frecuencia que el modo de punto de acceso. En la figura 1-33 se muestran ambos modos.

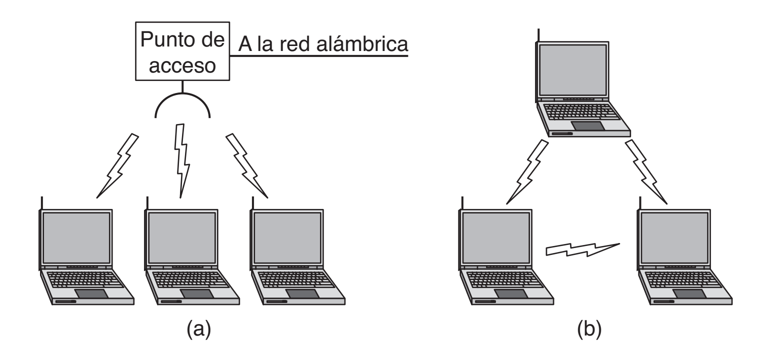


Figura 1-33. (a) Red inalámbrica con un punto de acceso. (b) Red *ad hoc*.

La transmisión 802.11 se complica debido a las condiciones inalámbricas que varían incluso con pequeños cambios en el entorno. En las frecuencias usadas para 802.11 las señales de radio pueden

rebotar de objetos sólidos, de modo que varios ecos de una transmisión podrían llegar a un receptor a través de distintas rutas. Los ecos se pueden cancelar o reforzar unos a otros y provocar que la señal recibida fluctúe de manera considerable. Este fenómeno se llama **desvanecimiento multitrayectoria** y se muestra en la figura 1-34.

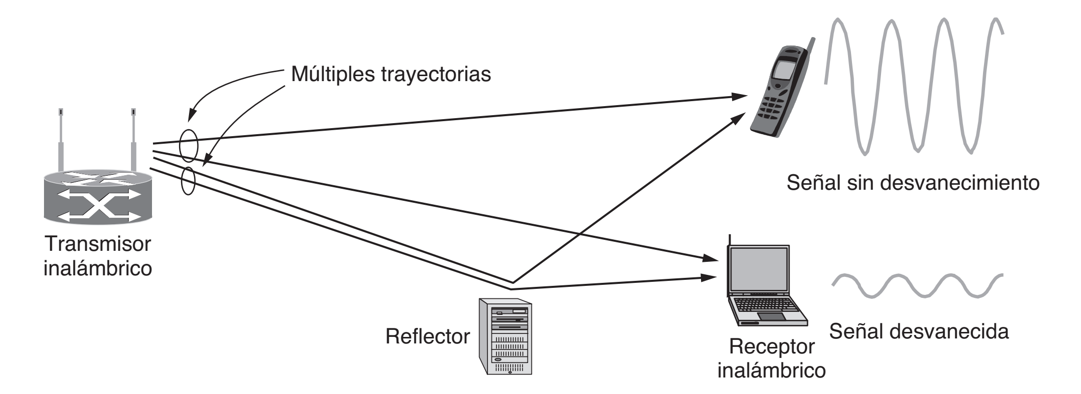


Figura 1-34. Desvanecimiento multitrayectorias.

La idea clave para solventar las condiciones inalámbricas variables es la **diversidad de rutas**, o el envío de información a través de múltiples rutas independientes. De esta forma, es probable que la información se reciba incluso si una de las rutas resulta ser pobre debido a un desvanecimiento. Por lo general estas rutas independientes están integradas al esquema de modulación digital en la capa física. Las opciones incluyen el uso de distintas frecuencias a lo largo de la banda permitida, en donde se siguen distintas rutas espaciales entre los distintos pares de antenas o se repiten bits durante distintos periodos.

Las distintas versiones de 802.11 han usado todas estas técnicas. El estándar inicial (1997) definió una LAN inalámbrica que podía operar a 1 Mbps o 2 Mbps mediante saltos entre frecuencias o también se podía extender la señal a lo largo del espectro permitido. Casi de inmediato surgieron las quejas de las personas diciendo que era muy lenta, por lo que se empezó a trabajar en estándares más veloces. El diseño de espectro extendido se amplió y convirtió en el estándar 802.11b (1999) que operaba a velocidades de hasta 11 Mbps. Los estándares 802.11a (1999) y 802.11g (2003) cambiaron a un esquema de modulación distinto llamado **OFDM** (**Multiplexado por División de Frecuencias Ortogonales**, del inglés *Orthogonal Frequency Division Multiplexing*). Este esquema divide una banda amplia de espectro en muchas fracciones estrechas, a través de las cuales se envían distintos bits en paralelo. Este esquema mejorado, que estudiaremos en el capítulo 2, logró aumentar las velocidades en bits de los estándares 802.11a/g hasta 54 Mbps. Es un aumento considerable, pero las personas querían una velocidad aún mayor para soportar usos más demandantes. La versión más reciente es 802.11n (2009), la cual utiliza bandas de frecuencia más amplias y hasta cuatro antenas por computadora para alcanzar velocidades de hasta 450 Mbps.

Como la tecnología inalámbrica es un medio de difusión por naturaleza, los radios 802.11 también tienen que lidiar con el problema de que las múltiples transmisiones que se envían al mismo tiempo tendrán colisiones, lo cual puede interferir con la recepción. Para encargarse de este problema, 802.11 utiliza un esquema **CSMA** (**Acceso Múltiple por Detección de Portadora**, del inglés *Carrier Sense Multiple Access*) basado en ideas provenientes de la Ethernet alámbrica que, irónicamente, se basó en una de las primeras redes inalámbricas desarrolladas en Hawai, llamada **ALOHA**. Las computadoras esperan durante un intervalo corto y aleatorio antes de transmitir, y difieren sus transmisiones si escuchan que hay alguien más transmitiendo. Este esquema reduce la probabilidad de que dos computadoras envíen datos al mismo tiempo, pero no funciona tan bien como en el caso de las computadoras conectadas por cables.

Para ver por qué, examine la figura 1-35. Suponga que la computadora *A* está transmitiendo datos a la computadora *B*, pero el rango de radio del transmisor de *A* es demasiado corto como para llegar a la computadora *C*. Si *C* desea transmitir a *B* puede escuchar antes de empezar, pero el hecho de que no escuche nada no significa que su transmisión vaya a tener éxito. La incapacidad de *C* de escuchar a *A* antes de empezar provoca algunas colisiones. Después de una colisión, el emisor espera durante un retardo aleatorio más largo y vuelve a transmitir el paquete. A pesar de ésta y de otras cuestiones, el esquema funciona bastante bien en la práctica.

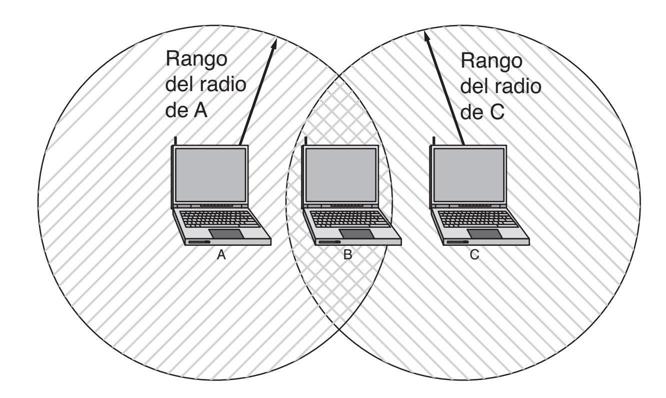


Figura 1-35. El rango de un solo radio tal vez no cubra todo el sistema.

Otro problema es la movilidad. Si un cliente móvil se aleja del punto de acceso que utiliza y entra en el rango de un punto de acceso distinto, se requiere alguna forma de entrega. La solución es que una red 802.11 puede consistir de múltiples celdas, cada una con su propio punto de acceso, y de un sistema de distribución que las conecte. Con frecuencia el sistema de distribución es Ethernet conmutada, pero puede usar cualquier tecnología. A medida que los clientes se desplazan, tal vez encuentren otro punto de acceso con una mejor señal que la que tienen en ese momento y pueden cambiar su asociación. Desde el exterior, el sistema completo se ve como una sola LAN alámbrica.

Aclarado el punto, la movilidad en el estándar 802.11 ha sido de un valor limitado si se le compara con la movilidad disponible en la red de telefonía móvil. Por lo general, el 802.11 lo utilizan los clientes nómadas que van de una ubicación fija a otra, en vez de usarlo en el camino. Estos clientes en realidad no necesitan movilidad. Incluso cuando se utiliza la movilidad que ofrece el estándar 802.11, se extiende sobre una sola red 802.11, que podría cubrir cuando mucho un edificio extenso. Los esquemas en lo futuro tendrán que proveer movilidad a través de distintas redes y diferentes tecnologías (por ejemplo, 802.21).

Por último tenemos el problema de la seguridad. Como las transmisiones inalámbricas son difundidas, es fácil que las computadoras cercanas reciban paquetes de información que no estaban destinados para ellas. Para evitar esto, el estándar 802.11 incluyó un esquema de cifrado conocido como **WEP** (**Privacidad Equivalente a Cableado**, del inglés *Wired Equivalent Privacy*). La idea era lograr que la seguridad inalámbrica fuera igual a la seguridad alámbrica. Es una buena idea, pero por desgracia el esquema era imperfecto y no pasó mucho tiempo para que fallara (Borisov y colaboradores, 2001). Desde entonces se reemplazó con esquemas más recientes que tienen distintos detalles criptográficos en el estándar 802.11i, conocido también como **Acceso protegido WiFi**, que en un principio se llamó **WPA** pero ahora se reemplazó por el **WPA2**.

El estándar 802.11 provocó una revolución en las redes inalámbricas que está destinada a continuar. Aparte de los edificios, se ha empezado a instalar en trenes, aviones, botes y automóviles de modo que las

personas puedan navegar por Internet en cualquier parte a donde vayan. Los teléfonos móviles y todo tipo de electrodomésticos, desde las consolas de juego hasta las cámaras digitales, se pueden comunicar con este estándar. En el capítulo 4 hablaremos detalladamente sobre este estándar.

1.5.3 Redes RFID y de sensores

Las redes que hemos estudiado hasta ahora están compuestas de dispositivos de cómputo fáciles de reconocer, desde computadoras hasta teléfonos móviles. Gracias a la **Identificación por Radio Frecuencia** (**RFID**), los objetos cotidianos también pueden formar parte de una red de computadoras.

Una etiqueta RFID tiene la apariencia de una calcomanía del tamaño de una estampilla postal que se puede pegar (o incrustar) en un objeto, de modo que se pueda rastrear. El objeto podría ser una vaca, un pasaporte o un libro. La etiqueta consiste en un pequeño microchip con un identificador único y una antena que recibe transmisiones por radio. Los lectores RFID instalados en puntos de rastreo encuentran las etiquetas cuando están dentro del rango y las interrogan para obtener su información como se muestra en la figura 1-36. Las aplicaciones incluyen: verificar identidades, administrar la cadena de suministro, carreras de sincronización y reemplazar códigos de barras.

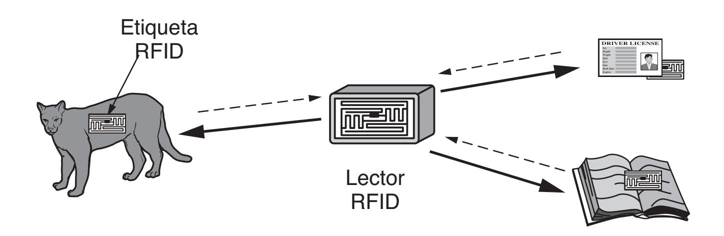


Figura 1-36. La tecnología RFID se utiliza para conectar objetos cotidianos en red.

Hay muchos tipos de RFID, cada uno con distintas propiedades, pero tal vez el aspecto más fascinante de la tecnología RFID sea que la mayoría de las etiquetas RFID no tienen enchufe eléctrico ni batería, sino que toda la energía necesaria para operarlos se suministra en forma de ondas de radio a través de los lectores RFID. A esta tecnología se le denomina **RFID pasiva** para diferenciarla de la **RFID activa** (menos común), en la cual hay una fuente de energía en la etiqueta.

La **RFID de UHF** (**RFID de Ultra Alta Frecuencia**, del inglés *Ultra-High Frequency RFID*) es una forma común de RFID que se utiliza en algunas licencias de conducir. Los lectores envían señales en la banda de 902-928 MHz en Estados Unidos. Las etiquetas se pueden comunicar a distancias de varios metros al cambiar la forma en que reflejan las señales de los lectores; el lector es capaz de recuperar estas reflexiones. A esta forma de operar se le conoce como **retrodispersión** (*backscatter*).

La **RFID de HF** (**RFID de Alta Frecuencia**, del inglés *High Frequency RFID*) es otro tipo popular de RFID que opera a 13.56 MHz y se utiliza por lo general en pasaportes, tarjetas de crédito, libros y sistemas de pago sin contacto. La RFID de HF tiene un rango corto, por lo común de un metro o menos, debido a que el mecanismo físico se basa en la inducción en vez de la retrodispersión. Existen también otras formas de RFID que utilizan otras frecuencias, como la **RFID de LF** (**RFID de Baja Frecuencia**, del inglés *Low Frequency RFID*) que se desarrolló antes de la RFID de HF y se utilizaba para rastrear animales. Es el tipo de RFID que podría llegar a estar en su gato.

Los lectores RFID deben resolver de alguna manera el problema de lidiar con varias etiquetas dentro del rango de lectura. Esto significa que una etiqueta no puede simplemente responder cuando escucha a un lector, o que puede haber colisiones entre las señales de varias etiquetas. La solución es similar a la

metodología aplicada en el estándar 802.11: las etiquetas esperan durante un intervalo corto y aleatorio antes de responder con su identificación, lo cual permite al lector reducir el número de etiquetas individuales e interrogarlas más.

La seguridad es otro problema. La habilidad de los lectores RFID de rastrear con facilidad un objeto, y por ende a la persona que lo utiliza, puede representar una invasión a la privacidad. Por desgracia es difícil asegurar las etiquetas RFID debido a que carecen del poder de cómputo y de comunicación requerido para ejecutar algoritmos criptográficos sólidos. En vez de ello se utilizan medidas débiles como las contraseñas (que se pueden quebrantar con facilidad). Si un oficial en una aduana puede leer de manera remota una tarjeta de identificación, ¿qué puede evitar que otras personas rastreen esa misma tarjeta sin que usted lo sepa? No mucho.

Las etiquetas RFID empezaron como chips de identificación, pero se están convirtiendo con rapidez en computadoras completas. Por ejemplo, muchas etiquetas tienen memoria que se puede actualizar y que podemos consultar después, de modo que se puede almacenar información sobre lo que ocurra con el objeto etiquetado. Reiback y colaboradores (2006) demostraron que esto significa que se aplican todos los problemas comunes del software malicioso de computadora, sólo que ahora sería posible usar su gato o su pasaporte para esparcir un virus de RFID.

La **red de sensores** va un paso más allá en cuanto a capacidad, en comparación con la RFID. Las redes de sensores se implementan para vigilar los aspectos del mundo físico. Hasta ahora se han utilizado en su mayor parte para la experimentación científica, como el monitoreo de los hábitats de las aves, la actividad volcánica y la migración de las cebras, pero es probable que pronto surjan aplicaciones para el cuidado de la salud, equipo de monitoreo de vibraciones y rastreo de artículos congelados, refrigerados u otro tipo de perecederos.

Los nodos sensores son pequeñas computadoras, por lo general del tamaño de un control de llave, que tienen sensores de temperatura, vibración y demás. Muchos nodos se colocan en el entorno que se va a vigilar. Por lo general tienen baterías, aunque también pueden obtener energía de las vibraciones del Sol. Al igual que la RFID, tener suficiente energía es un reto clave por lo que los nodos deben comunicarse con cuidado para transmitir la información de sus sensores a un punto externo de recolección. Una estrategia común es que los nodos se autoorganicen para transmitir mensajes unos de otros, como se muestra en la figura 1-37. Este diseño se conoce como **red multisaltos**.

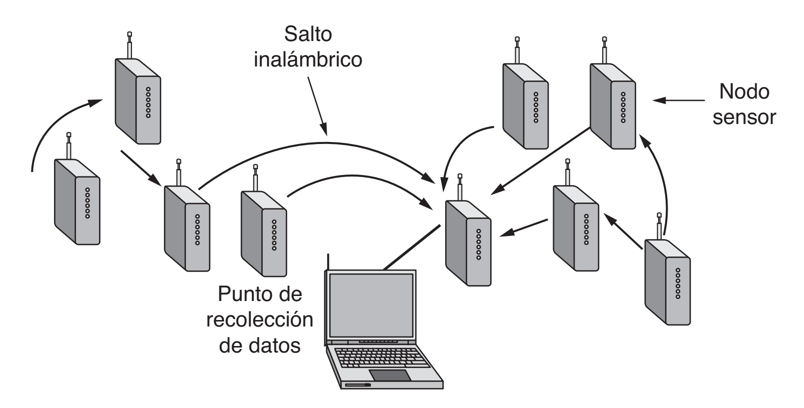


Figura 1-37. Topología multisaltos de una red de sensores.

Es probable que las redes RFID y de sensores sean mucho más capaces y dominantes en el futuro. Los investigadores ya han combinado lo mejor de ambas tecnologías al crear prototipos de etiquetas RFID con sensores de luz, movimiento y otros sensores (Sample y colaboradores, 2008).

1.6 ESTANDARIZACIÓN DE REDES

Existen muchos distribuidores y proveedores de servicios de red, cada uno con sus propias ideas de cómo hacer las cosas. Sin coordinación existiría un caos completo y los usuarios nunca lograrían hacer nada. La única salida es acordar ciertos estándares de redes. Los buenos estándares no sólo permiten que distintas computadoras se comuniquen, sino que también incrementan el mercado para los productos que se adhieren a estos estándares. Un mercado más grande conduce a la producción en masa, economías de escala en la fabricación, mejores implementaciones y otros beneficios que reducen el precio y aumentan más la aceptación.

En esta sección veremos las generalidades sobre el importante pero poco conocido mundo de la estandarización internacional. Pero primero hablaremos sobre lo que debe incluir un estándar. Una persona razonable podría suponer que un estándar nos dice cómo debe funcionar un protocolo, de modo que podamos hacer un buen trabajo al implementarlo. Esa persona estaría equivocada.

Los estándares definen lo que se requiere para la interoperabilidad y nada más. Esto permite que emerja un mercado más grande y también deja que las empresas compitan con base en qué tan buenos son sus productos. Por ejemplo, el estándar 802.11 define muchas velocidades de transmisión pero no dice cuándo un emisor debe utilizar cierta velocidad, lo cual es un factor clave para un buen desempeño. Esto queda a criterio del fabricante del producto. A menudo es difícil obtener una interoperabilidad de esta forma, ya que hay muchas opciones de implementación y los estándares por lo general definen muchas opciones. Para el 802.11 había tantos problemas que, en una estrategia que se convirtió en práctica común, un grupo llamado **Alianza WiFi** empezó a trabajar en la interoperabilidad con el estándar 802.11.

De manera similar, un estándar de protocolos define el protocolo que se va a usar a través del cable pero no la interfaz de servicio dentro de la caja, excepto para ayudar a explicar el protocolo. A menudo las interfaces de servicio reales son de marca registrada. Por ejemplo, la manera en que TCP hace interfaz con IP dentro de una computadora no importa para comunicarse con un host remoto. Sólo importa que el host remoto utilice TCP/IP. De hecho, TCP e IP se implementan juntos con frecuencia sin ninguna interfaz distinta. Habiendo dicho esto, las buenas interfaces de servicio (al igual que las buenas API) son valiosas para lograr que se utilicen los protocolos, además de que las mejores (como los sockets de Berkeley) se pueden volver muy populares.

Los estándares se dividen en dos categorías: de facto y de jure. Los estándares *de facto* (del latín “del hecho”) son aquellos que simplemente aparecieron, sin ningún plan formal. El protocolo HTTP con el que opera la web empezó como un estándar de facto. Era parte de los primeros navegadores WWW desarrollados por Tim Berners-Lee en CERN y su uso se popularizó debido al crecimiento de la web. Bluetooth es otro ejemplo. En un principio fue desarrollado por Ericsson, pero ahora todo el mundo lo utiliza.

En contraste, los estándares *de jure* (del latín “por ley”) se adoptan por medio de las reglas de alguna organización formal de estandarización. Por lo general las autoridades de estandarización internacionales se dividen en dos clases: las que se establecieron mediante un tratado entre gobiernos nacionales y las conformadas por organizaciones voluntarias que no surgieron de un tratado. En el área de los estándares de redes de computadoras hay varias organizaciones de cada tipo, en especial: ITU, ISO, IETF e IEEE, de las cuales hablaremos a continuación.

En la práctica, las relaciones entre los estándares, las empresas y los organismos de estándares son complicadas. A menudo los estándares de facto evolucionan para convertirse en estándares *de jure*, en especial si tienen éxito. Esto ocurrió en el caso de HTTP, que fue elegido rápidamente por el IETF. Es común que los organismos de estándares ratifiquen los estándares de otros organismos, dando la impresión de aprobarse unos a otros, en un esfuerzo por incrementar el mercado para una tecnología. En estos días, muchas alianzas de negocios *ad hoc* que se forman con base en tecnologías específicas también desempeñan un papel considerable en el desarrollo y refinamiento de los estándares de redes. Por ejemplo, **3GPP**

(**Proyecto de Sociedad de Tercera Generación**, del inglés *Third Generation Parthnership Project*) es una colaboración entre asociaciones de telecomunicaciones que controla los estándares de la telefonía móvil 3GS UMTS.

1.6.1 Quién es quién en el mundo de las telecomunicaciones

El estado legal de las compañías telefónicas del mundo varía de manera considerable de un país a otro. En un extremo se encuentra Estados Unidos, que tiene cerca de 200 compañías privadas telefónicas separadas (la mayoría muy pequeñas). Con la disolución de AT&T en 1984 (que en ese entonces era la corporación más grande del mundo que proveía servicio a cerca del 80% de los teléfonos en América) surgieron unas cuantas compañías más, junto con la Ley de Telecomunicaciones en 1996 que replanteó las reglamentaciones para fomentar la competencia.

Al otro extremo están los países en donde el gobierno nacional tiene un total monopolio sobre toda la comunicación, incluyendo el correo, telégrafo, teléfono y a menudo la radio y televisión. Una gran parte del mundo entra en esta categoría. En algunos casos la autoridad de telecomunicaciones es una compañía nacionalizada, y en otros es simplemente una rama del gobierno, por lo general conocida como **PTT** (**Oficina de Correos, Telegrafía y Teléfonos**, del inglés *Port, Telegraph & Telephone*). A nivel mundial la tendencia es ir hacia la liberalización y la competencia para alejarse del monopolio gubernamental. La mayoría de los países europeos han privatizado ya (en forma parcial) sus oficinas PTT, pero en otras partes el proceso apenas si va ganando fuerza lentamente.

Con todos estos diferentes proveedores de servicios, existe sin duda la necesidad de proveer compatibilidad a escala mundial para asegurar que las personas (y computadoras) en un país puedan llamar a sus contrapartes en otro país. En realidad, esta necesidad ha existido desde hace un buen tiempo. En 1865, los representantes de muchos gobiernos europeos se reunieron para formar el predecesor de lo que hoy es **ITU** (**Unión Internacional de Telecomunicaciones**, del inglés *International Telecommunication Union*). Su tarea era estandarizar las telecomunicaciones internacionales, que en esos días consistían en la telegrafía. Aun en ese entonces era evidente que si la mitad de los países utilizaban código Morse y la otra mitad utilizaban algún otro código, iba a haber problemas. Cuando el teléfono entró a dar servicio internacional, la ITU también se hizo cargo de la tarea de estandarizar la telefonía. En 1947 la ITU se convirtió en una agencia de las Naciones Unidas.

La ITU tiene cerca de 200 miembros gubernamentales, incluyendo casi todos los miembros de las Naciones Unidas. Como Estados Unidos no cuenta con una PTT, alguien más tuvo que representar a este país en la ITU. Esta tarea repercutió en el Departamento de Estado, probablemente con la justificación de que la ITU tenía que lidiar con países extranjeros, lo cual era la especialidad de este departamento. La ITU cuenta también con más de 700 miembros de sectores y asociados. Entre ellos se incluyen las compañías telefónicas (como AT&T, Vodafone, Sprint), los fabricantes de equipo de telecomunicaciones (como Cisco, Nokia, Nortel), los distribuidores de computadoras (como Microsoft, Agilent, Toshiba), los fabricantes de chips (como Intel, Motorola, TI) y demás compañías interesadas (como Boeing, CBS, VeriSign).

La ITU tiene tres sectores principales. Nos enfocaremos principalmente en **ITU-T**, Sector de estandarización de telecomunicaciones, que se encarga de los sistemas de telefonía y comunicaciones de datos. Antes de 1993 a este sector se le llamaba **CCITT**, siglas de su nombre en francés, Comité Consultatif International Telégraphique et Teléphonique. **ITU-R**, sector de radiocomunicaciones, se encarga de coordinar el uso de las radiofrecuencias a nivel mundial por parte de los grupos de interés competidores. El otro sector es ITU-D, sector de desarrollo que promueve el desarrollo de las tecnologías de información y comunicación para estrechar la “división digital” entre los países con acceso efectivo a las tecnologías de información y los países con acceso limitado.

La tarea del sector ITU-T es hacer recomendaciones técnicas sobre las interfaces de telefonía, telegrafía y comunicación de datos. A menudo estas recomendaciones se convierten en estándares con reconocimiento internacional, aunque técnicamente las recomendaciones son sólo sugerencias que los gobiernos pueden adoptar o ignorar según lo deseen (porque los gobiernos son como niños de 13 años; no les gusta recibir órdenes). En la práctica, un país que desee adoptar un estándar de telefonía distinto al utilizado por el resto del mundo tiene la libertad de hacerlo, pero es a costa de quedar aislado de todos los demás. Esto podría funcionar para Corea del Norte, pero en cualquier otra parte sería un verdadero problema.

El verdadero trabajo del sector ITU-T se lleva a cabo en sus **Grupos de estudio** (*Study Groups*, o **SG**). En la actualidad hay 10 grupos de estudio de hasta 400 personas cada uno, en donde se tratan temas que varían desde la facturación telefónica y los servicios multimedia hasta la seguridad. Por ejemplo, el SG 15 estandariza las tecnologías DSL que son muy populares para conectarse a Internet. Para que sea posible realizar su trabajo, los grupos de estudio se dividen en **Equipos de trabajo** (*Working Parties*), que a su vez se dividen en **Equipos de expertos** (*Expert Teams*), los que a su vez se dividen en grupos ad hoc. La burocracia siempre será burocracia.

A pesar de todo esto, el sector ITU-T realmente hace su trabajo. Desde su creación ha producido más de 3 000 recomendaciones, muchas de las cuales son de uso popular en la práctica. Por ejemplo, la recomendación H.264 (que también es un estándar de ISO conocido como MPEG-4 AVC) es muy utilizada para la compresión de video, y los certificados de claves públicas X.509 se utilizan para la navegación web segura y el correo con firma digital.

A medida que el campo de las telecomunicaciones completa la transición iniciada en la década de 1980 para dejar de ser totalmente nacional y pasar a ser totalmente global, los estándares serán cada vez más importantes y cada vez más organizaciones querrán involucrarse en el proceso de establecer estos estándares. Para obtener más información sobre la ITU, consulte a Irmer (1994).

1.6.2 Quién es quién en el mundo de los estándares internacionales

Los estándares internacionales son producidos por la **ISO** (**Organización Internacional de Estándares**, del inglés *International Standards Organization*†), una organización voluntaria no surgida de un tratado y fundada en 1946. Sus miembros son las organizaciones nacionales de estándares de los 157 países miembros. Entre estos miembros están ANSI (Estados Unidos), BSI (Inglaterra), AFNOR (Francia), DIN (Alemania) y otras 153 organizaciones más.

La ISO emite estándares sobre una gran variedad de temas, que varían desde tuercas y pernos (literalmente) hasta los recubrimientos de los postes telefónicos [sin mencionar los granos de cacao (ISO 2451), las redes de pescar (ISO 1530), la ropa interior femenina (ISO 4416) y muchos otros temas más que no parecieran estar sujetos a la estandarización]. En cuestiones de estándares de telecomunicaciones, la ISO y el ITU-T cooperan con frecuencia (ISO es miembro del ITU-T) para evitar la ironía de dos estándares internacionales oficiales y mutuamente incompatibles.

Se han emitido más de 17 000 estándares, incluyendo los estándares OSI. La ISO tiene más de 200 Comités Técnicos (TC) enumerados en el orden de su creación, cada uno trata un tema específico. El TC1 trata con las tuercas y tornillos (la estandarización de los pasos de rosca de los tornillos). El JTC1 trata con la tecnología de información, incluyendo las redes, las computadoras y el software. Es el primer (y hasta ahora el único) Comité Técnico unido, el cual se creó en 1987 al fusionar el TC97 con las actividades en el IEC, otro organismo de estandarización. Cada TC tiene subcomités (SC), los que a su vez se dividen en grupos de trabajo (WG).

† Para los puristas, el verdadero nombre de ISO es Organización Internacional para la Estandarización.

El verdadero trabajo se hace en gran parte en los WG a través de los más de 100 000 voluntarios en todo el mundo. Muchos de estos "voluntarios" se asignan para trabajar en cuestiones de la ISO por sus patrones, cuyos productos se están estandarizando. Otros voluntarios son funcionarios de gobierno interesados en que la forma en que se hacen las cosas en su país llegue a ser el estándar internacional. También participan expertos académicos en muchos de los WG.

El procedimiento que utiliza la ISO para adoptar estándares se ha diseñado para lograr un consenso tan amplio como sea posible. El proceso empieza cuando una de las organizaciones nacionales de estándares siente la necesidad de un estándar internacional en cierta área. Después se forma un grupo de trabajo para proponer un **CD** (**Borrador de Comité**, del inglés *Committee Draft*). Después se circula el CD a todos los miembros, quienes tienen seis meses para criticarlo. Si una mayoría considerable lo aprueba, se produce un documento revisado llamado **DIS** (**Borrador de Estándar Internacional**, del inglés *Draft International Standard*), y se circula para que los miembros comenten y voten. Con base en los resultados de esta ronda, se prepara, aprueba y publica el texto final del **IS** (**Estándar Internacional**, del inglés *International Standard*). En áreas de mucha controversia, tal vez un CD o DIS tenga que pasar por varias versiones antes de adquirir suficientes votos, y el proceso completo puede tardar años.

El **NIST** (**Instituto Nacional de Estándares y Tecnología**, del inglés *National Institute of Standards and Technology*) forma parte del Departamento de Comercio. Solía llamarse Oficina Nacional de Estándares. Este organismo emite estándares obligatorios para las compras hechas por el gobierno de Estados Unidos, excepto las que realiza el Departamento de Defensa, el cual define sus propios estándares.

Otro protagonista importante en el mundo de los estándares es el **IEEE** (**Instituto de Ingenieros Eléctricos y Electrónicos**, del inglés *Institute of Electrical and Electronics Engineers*), la organización profesional más grande del mundo. Además de publicar muchas revistas y organizar numerosas conferencias cada año, el IEEE tiene un grupo de estandarización que desarrolla parámetros en el área de la ingeniería eléctrica y la computación. El comité 802 del IEEE ha estandarizado muchos tipos de redes LAN. Más adelante en el libro estudiaremos algunos de sus logros. El verdadero trabajo se realiza a través de una colección de grupos de trabajo, los cuales se muestran en la figura 1-38. El índice de éxito de los diversos grupos de trabajo del comité 802 ha sido bajo; tener un número 802.x no es garantía de éxito. Aun así, el impacto de las historias exitosas (en especial 802.3 y 802.11) en la industria y el mundo ha sido enorme.

1.6.3 Quién es quién en el mundo de estándares de Internet

El amplio mundo de Internet tiene sus propios mecanismos de estandarización, muy distintos a los de ITU-T e ISO. Para resumir en forma burda la diferencia, podemos decir que las personas que van a las reuniones de estandarización de la ITU o la ISO usan trajes, mientras que las personas que van a las reuniones de estandarización de Internet usan jeans (excepto cuando se reúnen en San Diego, en donde usan pantalones cortos y camisetas).

Las reuniones de la ITU-T y la ISO están pobladas de oficiales corporativos y burócratas para quienes la estandarización es su trabajo. Consideran la estandarización como algo positivo y dedican sus vidas a ella. Por otra parte, las personas de Internet prefieren la anarquía como cuestión de principios. Sin embargo, con cientos de millones de personas, cada una se ocupa de sus propios asuntos, no puede haber mucha comunicación. Por ende, algunas veces se necesitan los estándares por más lamentables que sean. En este contexto, una vez David Clark, del MIT, hizo un, ahora famoso, comentario acerca de que la estandarización de Internet consistía en "consenso aproximado y código en ejecución".

Cuando se inició ARPANET, el DoD creó un comité informal para supervisarla. En 1983 el comité cambió su nombre a **IAB** (**Consejo de Actividades de Internet**, del inglés *Internet Activities Board*) y recibió una misión un poco más amplia: mantener a los investigadores involucrados con ARPANET e

Número	Tema
802.1	Generalidades y arquitectura de redes LAN.
802.2 ↓	Control de enlaces lógicos.
802.3 *	Ethernet.
802.4 ↓	Token bus (se utilizó brevemente en las plantas de producción).
802.5	Token ring (la aportación de IBM al mundo de las redes LAN).
802.6 ↓	Bus doble de cola distribuida (la primera red de área metropolitana).
802.7 ↓	Grupo asesor técnico sobre tecnologías de banda ancha.
802.8 †	Grupo asesor técnico sobre tecnologías de fibra óptica.
802.9 ↓	Redes LAN isocrónicas (para aplicaciones en tiempo real).
802.10 ↓	Redes LAN virtuales y seguridad.
802.11 *	Redes LAN inalámbricas (WiFi).
802.12 ↓	Prioridad de demanda (AnyLAN, de Hewlett-Packard).
802.13	Número de mala suerte; nadie lo quiso.
802.14 ↓	Módems de cable (extinto: un consorcio industrial llegó primero).
802.15 *	Redes de área personal (Bluetooth, Zigbee).
802.16 *	Banda ancha inalámbrica (WIMAX).
802.17	Anillo de paquete elástico.
802.18	Grupo asesor técnico sobre cuestiones regulatorias de radio.
802.19	Grupo asesor técnico sobre la coexistencia de todos estos estándares.
802.20	Banda ancha móvil inalámbrica (similar a 802.16e).
802.21	Entrega independiente de los medios (para recorrer las tecnologías).
802.22	Red de área regional inalámbrica.

Figura 1-38. Los grupos de trabajo 802. Los importantes están marcados con *. Los que están marcados con ↓ están en hibernación. El que está marcado con † se dio por vencido y se deshizo.

Internet apuntando más o menos en la misma dirección, una actividad parecida a controlar una manada de gatos. El significado de las siglas "IAB" se cambió más adelante a **Consejo de Arquitectura de Internet**.

Cada uno de los aproximadamente 10 miembros del IAB encabezó una fuerza de trabajo sobre algún aspecto de importancia. El IAB se reunió varias veces al año para comentar sobre los resultados y brindar retroalimentación al DoD y la NSF, quienes proporcionaban la mayor parte de los fondos en esa época. Cuando se necesitaba un estándar (por ejemplo, un nuevo algoritmo de enrutamiento), los miembros del IAB lo discutían y después anunciaban el cambio de manera que los estudiantes de licenciatura, quienes eran el corazón del esfuerzo de software, pudieran implementarlo. La comunicación se llevaba a cabo mediante una serie de informes técnicos llamados **RFC** (**Petición de Comentarios**, del inglés *Request For Comments*). Los RFC se guardan en línea y cualquiera que se interese en ellos puede obtenerlos en *www.ietf.org/rfc*. Se enumeran en orden cronológico de creación. En la actualidad existen más de 5 000. Nos referiremos a muchos RFC en este libro.

Para 1989 Internet había crecido tanto que este estilo altamente informal ya no era funcional. Para entonces muchos distribuidores ofrecían productos TCP/IP y no querían cambiarlos sólo porque los investigadores habían tenido una mejor idea. En el verano de 1989, el IAB se volvió a organizar. Los investigadores pasaron a la **IRTF** (**Fuerza de Trabajo de Investigación de Internet**, del inglés *Internet Research Task Force*), la cual se hizo subsidiaria del IAB, junto con la **IETF** (**Fuerza de Trabajo de Ingeniería de Internet**, del inglés *Internet Engineering Task Force*). El IAB se repobló con gente que representaba un rango más amplio de organizaciones, no sólo la comunidad de investigación. En un principio fue un grupo que se perpetuaba a sí mismo, pues sus miembros servían por un término de dos años y los nuevos miembros eran designados por los antiguos. Más tarde se creó la **Sociedad de Internet** (*Internet Society*), formada por gente interesada en Internet. Así, podemos comparar en cierto sentido a la Sociedad de Internet con la ACM o el IEEE, ya que está gobernada por administradores elegidos, quienes designan a los miembros de la IAB.

El objetivo de esta división era hacer que la IRTF se concentrara en investigaciones a largo plazo, mientras que la IETF se encargaba de los problemas de ingeniería a corto plazo. La IETF se dividió en grupos de trabajo, cada uno con un problema específico por resolver. En un principio los presidentes de estos grupos de trabajo se reunieron como un comité de conducción para dirigir los trabajos de ingeniería. Los temas del grupo de trabajo incluyen nuevas aplicaciones, información de usuarios, integración de OSI, enrutamiento y direccionamiento, seguridad, administración de redes y estándares. En un momento dado se llegaron a formar tantos grupos de trabajo (más de 70) que se agruparon en áreas, en donde presidentes de cada una se reunía como el comité de conducción.

Además se adoptó un proceso de estandarización más formal con base en los patrones de la ISO. Para convertirse en una **Propuesta de estándar**, la idea básica se debe explicar en un RFC y debe generar suficiente interés en la comunidad para justificar su consideración. Para avanzar a la etapa de **Borrador de estándar**, una implementación funcional se debe probar rigorosamente por al menos dos sitios independientes durante cuatro meses como mínimo. Si el IAB se convence de que la idea es buena y el software funciona, puede declarar que el RFC es un **Estándar de Internet**. Algunos estándares de Internet se han convertido en estándares del DoD (MIL-STD), los cuales son obligatorios para los proveedores del DoD.

En cuanto a los estándares de la web, el **Consorcio World Wide Web** (*W3C*) desarrolla protocolos y lineamientos para facilitar el crecimiento a largo plazo de la web. Es un consorcio industrial encabezado por Tim Berners-Lee que se estableció en 1994, cuando la web realmente había empezado a despegar. Ahora el W3C tiene más de 300 miembros de todo el mundo y ha producido más de 100 Recomendaciones W3C, como se les dice a sus estándares, que tratan sobre temas tales como HTML y la privacidad en la web.

1.7 UNIDADES MÉTRICAS

Para evitar cualquier confusión, vale la pena indicar de manera explícita que en este libro, al igual que en la ciencia computacional en general, se utilizan medidas métricas en vez de unidades inglesas tradicionales (el sistema *furlong-stone-fortnight*). En la figura 1-39 se muestran los principales prefijos métricos. Por lo general se abrevian con base en sus primeras letras, y las unidades mayores a 1 se escriben en mayúsculas (KB, MB, etc.). Una excepción (por razones históricas) es kbps para kilobits/segundo. Así, una línea de comunicación de 1 Mbps transmite 10^6 bits/segundo y un reloj de 100 pseg (o 100 ps) genera un tic cada 10^{-10} segundos. Como mili y micro empiezan con la letra "m", hubo que tomar una decisión. Por lo general, "m" se utiliza para mili y "μ" (la letra griega mu) para micro.

Exp.	Explícito	Prefijo	Exp.	Explícito	Prefijo
10^{-3}	0.001	mili	10^{3}	1 000	Kilo
10^{-6}	0.000001	micro	10^{6}	1 000 000	Mega
10^{-9}	0.000000001	nano	10^{9}	1 000 000 000	Giga
10^{-12}	0.000000000001	pico	10^{12}	1 000 000 000 000	Tera
10^{-15}	0.000000000000001	femto	10^{15}	1 000 000 000 000 000	Peta
10^{-18}	0.000000000000000001	atto	10^{18}	1 000 000 000 000 000 000	Exa
10^{-21}	0.000000000000000000001	zepto	10^{21}	1 000 000 000 000 000 000 000	Zetta
10^{-24}	0.000000000000000000000001	yocto	10^{24}	1 000 000 000 000 000 000 000 000	Yotta

Figura 1-39. Los principales prefijos métricos.

También vale la pena señalar que para medir los tamaños de memoria, disco, archivos y bases de datos, en la práctica común de la industria las unidades tienen significados ligeramente distintos. Así, kilo significa 2^{10} (1 024) en vez de 10^{3} (1 000), ya que las memorias son siempre una potencia de dos. Por ende, una memoria de 1 KB contiene 1 024 bytes, no 1 000 bytes. Observe también que se utiliza una letra “B” mayúscula que significa “bytes” (unidades de ocho bits), en vez de una “b” minúscula que significa “bits”. De manera similar, una memoria de 1 MB contiene 2^{20} (1 048 576) bytes, una memoria de 1 GB contiene 2^{30} (1 073 741 824) bytes y una base de datos de 1 TB contiene 2^{40} (1 099 511 627 776) bytes. Sin embargo, una línea de comunicación de 1 kbps transmite 1000 bits por segundo y una red LAN de 10 Mbps opera a 10 000 000 bits/segundo, ya que estas velocidades no son potencias de dos. Por desgracia, muchas personas tienden a mezclar estos dos sistemas, en especial con los tamaños de los discos. Para evitar ambigüedades, en este libro utilizaremos los símbolos KB, MB, GB y TB para 2^{10}, 2^{20}, 2^{30} y 2^{40} bytes, respectivamente, y los símbolos kbps, Mbps, Gbps y Tbps para 10^{3}, 10^{6}, 10^{9} y 10^{12} bits/segundo, respectivamente.

1.8 ESQUEMA DEL RESTO DEL LIBRO

Este libro trata tanto los principios como la práctica de las redes de computadoras. La mayor parte de los capítulos empiezan con una explicación de los principios relevantes, seguida de varios ejemplos que ilustran estos principios. Por lo general estos ejemplos se toman de Internet y de las redes inalámbricas tales como la red de telefonía móvil, ya que ambas son importantes y muy distintas. Donde sea necesario también se dan otros ejemplos.

El libro está estructurado de acuerdo con el modelo híbrido de la figura 1-23. A partir del capítulo 2 comenzaremos a subir por la jerarquía de protocolos, empezando desde los cimientos. Veremos algunos antecedentes en el campo de la comunicación de datos que cubren a los sistemas de transmisión alámbricos e inalámbricos. Este material se enfoca en cómo entregar la información a través de los canales físicos, aunque sólo cubriremos los aspectos de arquitectura, no los de hardware. También veremos varios ejemplos de la capa física, como la red pública de telefonía conmutada, la red de telefonía móvil y la red de televisión por cable.

Los capítulos 3 y 4 tratan sobre la capa de enlace de datos en dos partes. El capítulo 3 analiza el problema de cómo enviar paquetes a través de un enlace, incluyendo la detección y corrección de errores.

Analizaremos la tecnología DSL (que se utiliza para el acceso de banda ancha a Internet sobre líneas telefónicas) como un ejemplo real de un protocolo de enlace de datos.

En el capítulo 4 examinaremos la subcapa de acceso al medio. Ésta es la parte de la capa de enlace de datos que se encarga de cómo compartir un canal entre varias computadoras. Los ejemplos que veremos incluyen redes inalámbricas, como 802.11 y RFID, además de redes LAN alámbricas como Ethernet clásica. Aquí también veremos los switches de la capa de enlace que conectan redes LAN, como Ethernet conmutada.

El capítulo 5 trata sobre la capa de red, en especial el enrutamiento. Veremos muchos algoritmos de enrutamiento, tanto estáticos como dinámicos. Incluso aunque existan buenos algoritmos de enrutamiento, si existe más tráfico del que la red pueda manejar, algunos paquetes se retrasarán o desecharán. Explicaremos esta cuestión, desde cómo evitar la congestión hasta cómo garantizar cierta calidad de servicio. Al conectar redes heterogéneas entre sí para formar interredes también se producen numerosos problemas, de los que hablaremos aquí. Además explicaremos con detalle la capa de red en Internet.

El capítulo 6 trata acerca de la capa de transporte. Daremos mucho énfasis a los protocolos orientados a conexión y la confiabilidad, ya que muchas aplicaciones los necesitan. Explicaremos también con detalle los protocolos de transporte de Internet: UDP y TCP, junto con sus aspectos de rendimiento.

El capítulo 7 se encarga de la capa de aplicación, sus protocolos y aplicaciones. El primer tema es DNS, que es el directorio telefónico de Internet. Después hablaremos sobre el correo electrónico, incluyendo una explicación de sus protocolos. Luego pasaremos a la web, con explicaciones detalladas del contenido estático y dinámico, además de lo que ocurre en los lados cliente y servidor. Más tarde analizaremos la multimedia en red, incluyendo audio y video de flujo continuo. Por último hablaremos sobre las redes de entrega de contenido, incluyendo la tecnología de igual a igual.

El capítulo 8 habla sobre la seguridad en las redes. Este tema tiene aspectos que se relacionan con todas las capas, por lo que es más fácil tratarlo después de haber explicado todas las capas a detalle. El capítulo empieza con una introducción a la criptografía. Después muestra cómo se puede utilizar la criptografía para garantizar la seguridad en las comunicaciones, el correo electrónico y la web. El capítulo termina con una explicación de algunas áreas en las que la seguridad choca con la privacidad, la libertad de expresión, la censura y otras cuestiones sociales.

El capítulo 9 contiene una lista con anotaciones de las lecturas sugeridas, ordenadas por capítulos. El objetivo es ayudar a los lectores que desean llevar más allá su estudio de las redes. Ese capítulo también incluye una bibliografía alfabética de todas las referencias citadas en este libro.

El sitio web de los autores en Pearson tiene una página con vínculos a muchos tutoriales, preguntas frecuentes (FAQ), compañías, consorcios industriales, organizaciones profesionales, organizaciones de estándares, tecnologías, documentos y demás.

1.9 RESUMEN

Las redes de computadoras tienen muchos usos, tanto para empresas como para individuos, en el hogar y en movimiento. Las empresas usan redes de computadoras para compartir la información corporativa, por lo general mediante el modelo cliente-servidor en donde las computadoras de los empleados actúan como clientes que acceden a poderosos servidores en la sala de máquinas. Para los individuos, las redes ofrecen acceso a una variedad de recursos de información y entretenimiento, así como una manera de comprar y vender productos y servicios. Con frecuencia los individuos acceden a Internet por medio de sus

proveedores de teléfono o cable en el hogar, aunque cada vez se utiliza más el acceso inalámbrico para laptops y teléfonos. Los avances tecnológicos permiten nuevos tipos de aplicaciones móviles y redes con computadoras integradas a los electrodomésticos y demás dispositivos para el consumidor. Los mismos avances generan cuestiones sociales tales como las relacionadas con la privacidad.

En términos generales, podemos dividir a las redes en LAN, MAN, WAN e interredes. Por lo general las redes LAN cubren todo un edificio y operan a velocidades altas. Las redes MAN comúnmente cubren toda una ciudad. El sistema de televisión por cable es un ejemplo, ya que ahora muchas personas lo utilizan para acceder a Internet. Las redes WAN pueden cubrir un país o continente. Algunas de las tecnologías utilizadas para construir estas redes son de punto a punto (como un cable), mientras que otras son de difusión (como las redes inalámbricas). Las redes se pueden interconectar con enrutadores para formar interredes, de las cuales Internet es el ejemplo más grande y popular. Las redes inalámbricas, como las redes LAN 802.11 y de telefonía móvil 3G, también se están volviendo muy populares.

El software de red se basa en los protocolos, que son reglas mediante las cuales los procesos se comunican entre sí. La mayoría de las redes soportan jerarquías de protocolos, en donde cada capa proporciona servicios a la capa inmediata superior y los aísla de los detalles sobre los protocolos que se utilizan en las capas inferiores. Por lo general las pilas de protocolos se basan en el modelo OSI o en el modelo TCP/IP. Ambos modelos tienen capas de enlace, red, transporte y aplicación, pero difieren en las otras capas. Los aspectos de diseño incluyen: confiabilidad, asignación de recursos, crecimiento, seguridad, etcétera. Gran parte de este libro se enfoca en los protocolos y su diseño.

Las redes proveen varios servicios a sus usuarios. Estos servicios pueden variar, desde la entrega de paquetes sin conexión de mejor esfuerzo hasta la entrega garantizada orientada a conexión. En algunas redes se proporciona servicio sin conexión en una capa y servicio orientado a conexión en la capa inmediata superior.

Entre las redes más conocidas están: Internet, la red de telefonía móvil 3G y las redes LAN 802.11. Internet evolucionó de ARPANET, a la que se agregaron otras redes para formar una interred. En realidad la Internet de la actualidad es una colección de muchos miles de redes que utilizan la pila de protocolos TCP/IP. La red de telefonía móvil 3G proporciona acceso inalámbrico y móvil a Internet, con velocidades de varios Mbps; además transmite llamadas de voz. Las redes LAN inalámbricas basadas en el estándar IEEE 802.11 se implementan en muchos hogares y cafés; pueden proporcionar conectividad a velocidades en mayores 100 Mbps. También están surgiendo nuevos tipos de redes, como las redes de sensores integradas y las redes basadas en tecnología RFID.

Para permitir que varias computadoras se comuniquen entre sí se requiere una gran cantidad de estandarización, tanto en hardware como en software. Las organizaciones tales como ITU-T, ISO, IEEE e IAB administran distintas partes del proceso de estandarización.

PROBLEMAS

1. Imagine que entrenó a Bernie, su perro San Bernardo, para que transporte una caja de tres cintas de 8 mm en vez de un termo con brandy (cuando se llene su disco, puede considerarlo una emergencia). Cada una de estas cintas contiene 7 gigabytes. El perro puede viajar a donde quiera que vaya, a una velocidad de 18 km/h. ¿Para qué rango de distancias tiene Bernie una velocidad mayor de datos que una línea de transmisión cuya velocidad de datos (sin sobrecarga) es de 150 Mbps? ¿Cómo cambiaría su respuesta si (i) se duplica la velocidad de Bernie; (ii) se duplica la capacidad de cada cinta; (iii) se duplica la velocidad de datos de la línea de transmisión?

2. Una alternativa a una LAN es simplemente un gran sistema de tiempo compartido con terminales para los usuarios. Cite dos ventajas de un sistema cliente-servidor que utiliza una LAN.

3. El rendimiento de un sistema cliente-servidor se ve muy influenciado por dos características principales de las redes: el ancho de banda de la red (es decir, cuántos bits/segundo puede transportar) y la latencia (cuántos segundos tarda el primer bit en viajar del cliente al servidor). Cite un ejemplo de una red que cuente con un ancho de banda alto pero también alta latencia. Después mencione un ejemplo de una red que tenga un ancho de banda bajo y una baja latencia.
4. Además del ancho de banda y la latencia, ¿qué otro parámetro se necesita para tener una buena caracterización de la calidad del servicio ofrecido por una red que se utiliza para:

   (i) tráfico de voz digitalizada?

   (ii) tráfico de video?

   (iii) tráfico de transacciones financieras?
5. Un factor en el retardo de un sistema de conmutación de paquetes de almacenamiento y envío es cuánto tiempo se requiere para almacenar y enviar un paquete a través de un switch. Si el tiempo de conmutación es de 10 μseg, ¿es probable que sea un factor importante en la respuesta de un sistema cliente-servidor en donde el cliente está en Nueva York y el servidor en California? Asuma que la velocidad de propagación en cobre y fibra óptica es de 2/3 la velocidad de la luz en el vacío.
6. Un sistema cliente-servidor utiliza una red satelital, en donde el satélite está a una altura de 40 000 km. ¿Cuál es el retardo en respuesta a una solicitud en el mejor de los casos?
7. En el futuro, cuando todos tengan una terminal casera conectada a una red de computadoras, serán posibles los consultas públicas instantáneas sobre asuntos legislativos pendientes. En algún momento las legislaturas existentes se podrían eliminar para dejar que el deseo del pueblo se exprese de manera directa. Los aspectos positivos de tal democracia directa son bastante obvios; comente sobre algunos de los aspectos negativos.
8. Cinco enrutadores se van a conectar a una subred de punto a punto. Entre cada par de enrutadores, los diseñadores pueden colocar una línea de alta velocidad, una de velocidad media, una de baja velocidad o ninguna línea. Si se requieren 100 ms de tiempo de la computadora para generar e inspeccionar cada topología, ¿cuánto tiempo se requiere para inspeccionarlas todas?
9. Una desventaja de una subred de difusión es la capacidad que se desperdicia cuando varios hosts tratan de acceder al canal al mismo tiempo. Como ejemplo simplista, suponga que el tiempo se divide en porciones (ranuras) discretas y que cada uno de los n hosts trata de usar el canal con una probabilidad de p durante cada porción de tiempo. ¿Qué fracción de las porciones se desperdiciará debido a las colisiones?
10. ¿Cuáles son dos razones para usar protocolos en capas? ¿Cuál es una posible desventaja de usar protocolos en capas?
11. A la presidenta de la empresa Specialty Paint Corp. se le ocurre la idea de trabajar con un fabricante de cerveza local para producir una lata de cerveza invisible (como medida para reducir la basura). La presidenta ordena a los de su departamento legal que investiguen el asunto; ellos a su vez piden ayuda al departamento de ingeniería. Como resultado, el ingeniero en jefe llama a su homólogo en la compañía de cerveza para discutir los aspectos técnicos del proyecto. Después los ingenieros se reportan con sus respectivos departamentos legales, quienes entonces conversan por teléfono para arreglar los aspectos legales. Por último, los dos presidentes corporativos discuten la cuestión financiera del trato. ¿Qué principio de un protocolo multicapas viola este mecanismo de comunicación, en el sentido del modelo OSI?
12. Cada una de dos redes proporciona un servicio confiable orientado a la conexión. Una de ellas ofrece un flujo de bytes confiable y la otra un flujo de mensajes confiable. ¿Son las dos redes idénticas? De ser así, ¿por qué se hace la distinción? Si no es así, mencione un ejemplo de cómo difieren.
13. ¿Qué significa “negociación” al hablar sobre protocolos de red? Cite un ejemplo.
14. En la figura 1-19 se muestra un servicio. ¿Hay algún otro servicio implícito en esta figura? Si es así, ¿en dónde está? En caso contrario, ¿por qué no?
15. En algunas redes, la capa de enlace de datos se encarga los errores de transmisión pidiendo que se retransmitan las tramas dañadas. Si la probabilidad de que una trama se dañe es p, ¿cuál es la cantidad promedio de transmisiones requeridas para enviar una trama? Suponga que las confirmaciones de recepción nunca se pierden.

16. Un sistema tiene una jerarquía de protocolos de *n* capas. Las aplicaciones generan mensajes con una longitud de *M* bytes. En cada una de las capas se agrega un encabezado de *h* bytes. ¿Qué fracción del ancho de banda de la red se llena con encabezados?

17. ¿Cuál es la principal diferencia entre TCP y UDP?

18. La subred de la figura 1-25(b) se diseñó para soportar una guerra nuclear. ¿Cuántas bombas se requerirían para particionar los nodos en dos conjuntos desconectados? Suponga que una bomba destruye a un nodo junto con todos los enlaces conectados a él.

19. Internet duplica su tamaño aproximadamente cada 18 meses. Aunque en realidad nadie lo sabe con certeza, alguien estimó que el número de hosts que incluía era de 600 millones en 2009. Use estos datos para calcular el número esperado de hosts de Internet para 2018. ¿Cree usted esto? Explique por qué si o por qué no.

20. Al transferir un archivo entre dos computadoras, hay dos estrategias de confirmación de recepción posibles. En la primera, el archivo se divide en paquetes y el receptor envía una confirmación de recepción por cada paquete individual, pero no envía una confirmación de recepción para la transferencia del archivo como un todo. En la segunda no se envía una confirmación de recepción para cada paquete individual, sino que se envía una confirmación de recepción de todo el archivo completo cuando llega. Comente sobre las dos estrategias.

21. Los operadores de redes de telefonía móvil necesitan saber en dónde se encuentran los teléfonos móviles (y sus usuarios). Explique por qué esto es malo para los usuarios. Ahora mencione las razones por las que esto es bueno para los usuarios.

22. ¿Qué tan largo era un bit en el estándar 802.3 original en metros? Use una velocidad de transmisión de 10 Mbps y suponga que la velocidad de propagación en cable coaxial es de 2/3 la velocidad de la luz en el vacío.

23. Una imagen tiene $1\,600 \times 1\,200$ píxeles con 3 bytes/píxel. Suponga que no está comprimida. ¿Cuánto tiempo tarda en transmitirse a través de un canal de modem de 56 kbps? ¿A través de un módem de cable de 1 Mbps? ¿A través de una red Ethernet de 10 Mbps? ¿A través de una red Ethernet de 100 Mbps? ¿A través de una red Gigabit Ethernet de 1 gbps?

24. Ethernet y las redes inalámbricas tienen ciertas similitudes y diferencias. Una propiedad de Ethernet es que sólo se puede transmitir una trama a la vez. ¿Comparte la red 802.11 esta propiedad con Ethernet? Explique su respuesta.

25. Mencione dos ventajas y dos desventajas de tener estándares internacionales para los protocolos de red.

26. Cuando un sistema tiene una parte permanente y una removible (como una unidad de CD-ROM y el CD-ROM), es importante que el sistema esté estandarizado de manera que distintas empresas puedan fabricar tanto las partes permanentes como las removibles y que todo pueda funcionar en conjunto. Cite tres ejemplos fuera de la industria de las computadoras en donde existan dichos estándares internacionales. Ahora cite tres áreas fuera de la industria de las computadoras en donde no existan.

27. Suponga que se cambian los algoritmos utilizados para implementar las operaciones en la capa *k*. ¿Cómo puede afectar esto a las operaciones en las capas $k - 1$ y $k + 1$?

28. Suponga que hay un cambio en el servicio (conjunto de operaciones) ofrecido por la capa *k*. ¿Cómo afecta esto a los servicios en las capas $k - 1$ y $k + 1$?

29. Proporcione una lista de razones por las que el tiempo de respuesta de un cliente puede ser mayor que el retardo en el mejor de los casos.

30. ¿Cuáles son las desventajas de usar celdas pequeñas de longitud fija en ATM?

31. Haga una lista de actividades que realiza a diario en donde se utilicen redes de computadoras. ¿Cómo se alteraría su vida si de repente se apagaran estas redes?

32. Averigüe qué redes se utilizan en su escuela o lugar de trabajo. Describa los tipos de redes, las topologías y los métodos de conmutación utilizados.

33. El programa *ping* le permite enviar un paquete de prueba a una ubicación dada para ver cuánto tarda en llegar hasta allá y regresar. Pruebe a usar *ping* para ver cuánto tiempo se requiere para ir de su ubicación hasta varias ubicaciones conocidas. Con base en estos datos, trace el tiempo de tránsito de una sola dirección a través de

Internet en función de la distancia. Lo más adecuado es utilizar universidades, ya que la ubicación de sus servidores se conoce con mucha precisión. Por ejemplo, *berkeley.edu* está en Berkeley, California; *mit.edu* está en Cambridge, Massachusetts; *vu.nl* está en Amsterdam, Holanda; *www.usyd.edu.au* está en Sydney, Australia; y *www.uct.ac.za* está en Cape Town, Sudáfrica.

34. Vaya al sitio web del IETF, *www.ietf.org*, para ver lo que están haciendo. Elija el proyecto que desee y escriba un informe de media página sobre el problema y la solución propuesta.

35. Internet está compuesta de una gran cantidad de redes. Su arreglo determina la topología de Internet. Hay una cantidad considerable de información en línea sobre la topología de Internet. Use un motor de búsqueda para averiguar más sobre la topología de Internet y escriba un breve informe con una síntesis de sus hallazgos.

36. Busque en Internet algunos de los puntos de interconexión importantes que se utilizan para encaminar paquetes en Internet en la actualidad.

37. Escriba un programa que implemente el flujo de mensajes desde la capa más alta hasta la capa más baja del modelo de protocolos de siete capas. Su programa debe incluir una función de protocolo separada para cada capa. Los encabezados de los protocolos son secuencias de hasta 64 caracteres. Cada función de protocolo tiene dos parámetros: un mensaje que se pasa del protocolo de la capa superior (un búfer de caracteres) y el tamaño del mensaje. Esta función añade su encabezado al frente del mensaje, imprime el nuevo mensaje en la salida estándar y después invoca a la función de protocolo del protocolo de la capa inferior. La entrada del programa es un mensaje de aplicación (una secuencia de 80 caracteres o menos).

2

LA CAPA FÍSICA

En este capítulo analizaremos la capa más baja en nuestro modelo de protocolos: la capa física. Ésta define las interfaces eléctricas, de temporización y demás interfaces mediante las cuales se envían los bits como señales a través de los canales. La capa física es la base sobre la cual se construye la red. Las propiedades de distintos tipos de canales físicos determinan el desempeño (por ejemplo, rendimiento, latencia y tasa de error), por lo que es un buen lugar para empezar nuestro viaje hacia la tierra de las redes.

Empezaremos con un análisis teórico de la transmisión de datos, sólo para descubrir que la Madre Naturaleza nos impuso ciertos límites en lo que se puede enviar a través de un canal. Después veremos tres tipos de medios de transmisión: guiados (cable de cobre y fibra óptica), inalámbricos (radio terrestre) y satélite. Cada una de estas tecnologías tiene distintas propiedades que afectan el diseño y el desempeño de las redes que las utilizan. Este material proveerá los antecedentes clave sobre las tecnologías de transmisión que se utilizan en las redes actuales.

Después conoceremos la modulación digital, que explica todo sobre la forma en que las señales analógicas se convierten en bits digitales y viceversa. Más adelante analizaremos los esquemas de multiplexión y exploraremos cómo se pueden colocar varias conversaciones en el mismo medio de transmisión al mismo tiempo, sin que unas interfieran con otras.

Por último, veremos tres ejemplos de sistemas de comunicación utilizados en la práctica para las redes de computadoras de área amplia: el sistema telefónico (fijo), el de telefonía móvil y el de televisión por cable. Todos estos sistemas son importantes en la práctica, por lo que dedicaremos tiempo a cada uno.

2.1 BASES TEÓRICAS PARA LA COMUNICACIÓN DE DATOS

Mediante la variación de alguna propiedad física, como el voltaje o la corriente, es posible transmitir información a través de cables. Si representamos el valor de este voltaje o corriente

como una función simple del tiempo, *f(t)*, podemos modelar el comportamiento de la señal y analizarlo matemáticamente. Este análisis es el tema de las siguientes secciones.

2.1.1 Análisis de Fourier

A principios del siglo XIX, el matemático francés Jean-Baptiste Fourier demostró que cualquier función periódica de comportamiento razonable, *g(t)* con un periodo *T*, se puede construir como la suma de un número (posiblemente infinito) de senos y cosenos:

$$g(t) = \frac{1}{2}c + \sum_{n=1}^{\infty} a_n \,\text{sen}(2\pi nft) + \sum_{n=1}^{\infty} b_n \cos(2\pi nft) \tag{2-1}$$

en donde $f = 1/T$ es la frecuencia fundamental, a_n y b_n son las amplitudes de seno y coseno del *n*-ésimo **armónico** (término) y *c* es una constante. A dicha descomposición se le denomina **serie de Fourier**. Podemos reconstruir la función a partir de la serie de Fourier. Esto es, si se conoce el periodo *T* y se dan las amplitudes, podemos encontrar la función original del tiempo realizando las sumas de la ecuación (2-1).

Es posible manejar una señal de datos con una duración finita (todas la tienen) con sólo imaginar que el patrón completo se repite una y otra vez de manera indefinida (es decir, el intervalo de *T* a *2T* es el mismo que de 0 a *T*, etcétera).

Podemos calcular las amplitudes a_n para cualquier $g(t)$ si multiplicamos ambos lados de la ecuación (2-1) por $\text{sen}(2\pi kft)$ y después integramos de 0 a *T*. Dado que

$$\int_0^T \text{sen}(2\pi kft)\,\text{sen}(2\pi nft)\,dt = \begin{cases} 0 \text{ para } k \neq n \\ T/2 \text{ para } k = n \end{cases}$$

sólo sobrevive un término de la sumatoria: a_n. La sumatoria b_n se desvanece por completo. De forma similar, si multiplicamos la ecuación (2-1) por $\cos(2\pi kft)$ e integramos entre 0 y *T*, podemos derivar b_n. Con sólo integrar ambos lados de la ecuación como está, podemos encontrar *c*. Los resultados de realizar estas operaciones son:

$$a_n = \frac{2}{T}\int_0^T g(t)\,\text{sen}(2\pi nft)\,dt \qquad b_n = \frac{2}{T}\int_0^T g(t)\cos(2\pi nft)\,dt \qquad c = \frac{2}{T}\int_0^T g(t)\,dt$$

2.1.2 Señales de ancho de banda limitado

La relevancia de todo esto para la comunicación de datos es que los canales reales afectan a las distintas señales de frecuencia de manera diferente. Consideremos un ejemplo específico: la transmisión del carácter ASCII “b” codificado en un byte de 8 bits. El patrón de bits que transmitirá es 01100010. La parte izquierda de la figura 2-1(a) muestra la salida de voltaje producido por la computadora transmisora. El análisis de Fourier de esta señal produce los siguientes coeficientes:

$$a_n = \frac{1}{\pi n}[\cos(\pi n/4) - \cos(3\pi n/4) + \cos(6\pi n/4) - \cos(7\pi n/4)]$$

$$b_n = \frac{1}{\pi n}[\text{sen}(3\pi n/4) - \text{sen}(\pi n/4) + \text{sen}(7\pi n/4) - \text{sen}(6\pi n/4)]$$

$$c = 3/4$$

En el lado derecho de la figura 2-1(a) se muestran las amplitudes de raíz cuadrada media, $\sqrt{a_n^2 + b_n^2}$ para los primeros términos. Estos valores son de interés debido a que sus cuadrados son proporcionales a la energía que se transmite en la frecuencia correspondiente.

Ninguna instalación transmisora puede enviar señales sin perder cierta potencia en el proceso. Si todos los componentes de Fourier disminuyeran en la misma proporción, la señal resultante se reduciría en amplitud pero no se distorsionaría [es decir, tendría la misma forma cuadrada de la figura 2-1(a)]. Por desgracia, todas las instalaciones transmisoras disminuyen los componentes de Fourier en distinto grado y, en consecuencia, introducen distorsión. Por lo general, las amplitudes se transmiten en su mayoría sin ninguna disminución en un cable, desde cero hasta cierta frecuencia f_c [se mide en ciclos/segundo o Hertz (Hz)], y se atenúan todas las frecuencias que están por encima de esta frecuencia de corte. El rango de frecuencia que se transmite sin una atenuación considerable se denomina **ancho de banda**. En la práctica, el corte en realidad no es muy abrupto, por lo que a menudo el ancho de banda referido es desde cero hasta la frecuencia a la que disminuyó la potencia recibida a la mitad.

El ancho de banda es una propiedad física del medio de transmisión que depende; por ejemplo, de la construcción, el grosor y la longitud de un cable o fibra óptica. A menudo se utilizan filtros para limitar el ancho de banda de una señal. Por ejemplo, los canales inalámbricos 802.11 pueden utilizar aproximadamente 20 MHz, por lo que los radios 802.11 filtran el ancho de banda de la señal con base en este tamaño. Otro ejemplo, los canales de televisión tradicionales (analógicos) ocupan 6 MHz cada uno, en un cable o a través del aire. Este filtrado permite que más señales compartan una región dada de un espectro, lo cual mejora la eficiencia del sistema en general. Lo que significa que el rango de frecuencia para ciertas señales no empezará en cero, pero no importa. El ancho de banda sigue siendo el rango de la banda de frecuencias que se transmiten, y la información que se puede transportar depende sólo de este ancho de banda, no de su frecuencia inicial ni final. Las señales que van desde cero hasta una frecuencia máxima se llaman señales de **banda base**. Las que se desplazan para ocupar un rango de frecuencias más altas, como es el caso de todas las transmisiones inalámbricas, se llaman señales de **pasa-banda**.

Ahora consideremos cómo luciría la señal de la figura 2-1(a) si el ancho de banda fuera tan pequeño que sólo se transmitieran las frecuencias más bajas [es decir, que la función se aproximara mediante los primeros términos de la ecuación (2-1)]. La figura 2-1(b) muestra la señal que resulta de un canal que sólo permite el paso del primer armónico (la fundamental, f). De manera similar, las figuras 2-1(c)-(e) muestran los espectros y las funciones reconstruidas para canales de mayor ancho de banda. Para la transmisión digital, el objetivo es recibir una señal con la suficiente fidelidad como para poder reconstruir la secuencia de bits que se envió. Se puede hacer esto con facilidad en la figura 2-1(e), por lo que sería un desperdicio usar más armónicos para recibir una réplica más precisa.

Si tenemos una tasa de bits de b bits/seg, el tiempo requerido para enviar los 8 bits en nuestro ejemplo, 1 bit a la vez, es de $8/b$ segundos, por lo que la frecuencia del primer armónico de esta señal es $b/8$ Hz. Una línea telefónica común, a menudo llamada **línea con calidad de voz**, tiene una frecuencia de corte introducida en forma artificial ligeramente mayor a 3 000 Hz. Esta restricción significa que el número de armónicos más altos que puede pasar es de aproximadamente 3 000/(b/8), o 24 000/b (el corte no es muy abrupto).

Para algunas tasas de datos, los números resultan como se muestra en la figura 2-2. De estos números queda claro que tratar de transmitir a 9 600 bps a través de una línea telefónica con calidad de voz, transformará la figura 2-1(a) en algo parecido a la figura 2-1(c), lo cual dificultará la recepción precisa del flujo de bits original. Debe ser obvio que a tasas de datos mucho mayores que 38.4 kbs no hay esperanza alguna para las señales *binarias*, aun cuando la instalación transmisora se encuentre totalmente libre de ruidos. En otras palabras, al limitar el ancho de banda se limita la tasa de datos, incluso en canales perfectos. Sin embargo, existen esquemas de codificación que utilizan diferentes niveles de voltaje y logran tasas de datos más altas. Más adelante en este capítulo veremos estos esquemas.

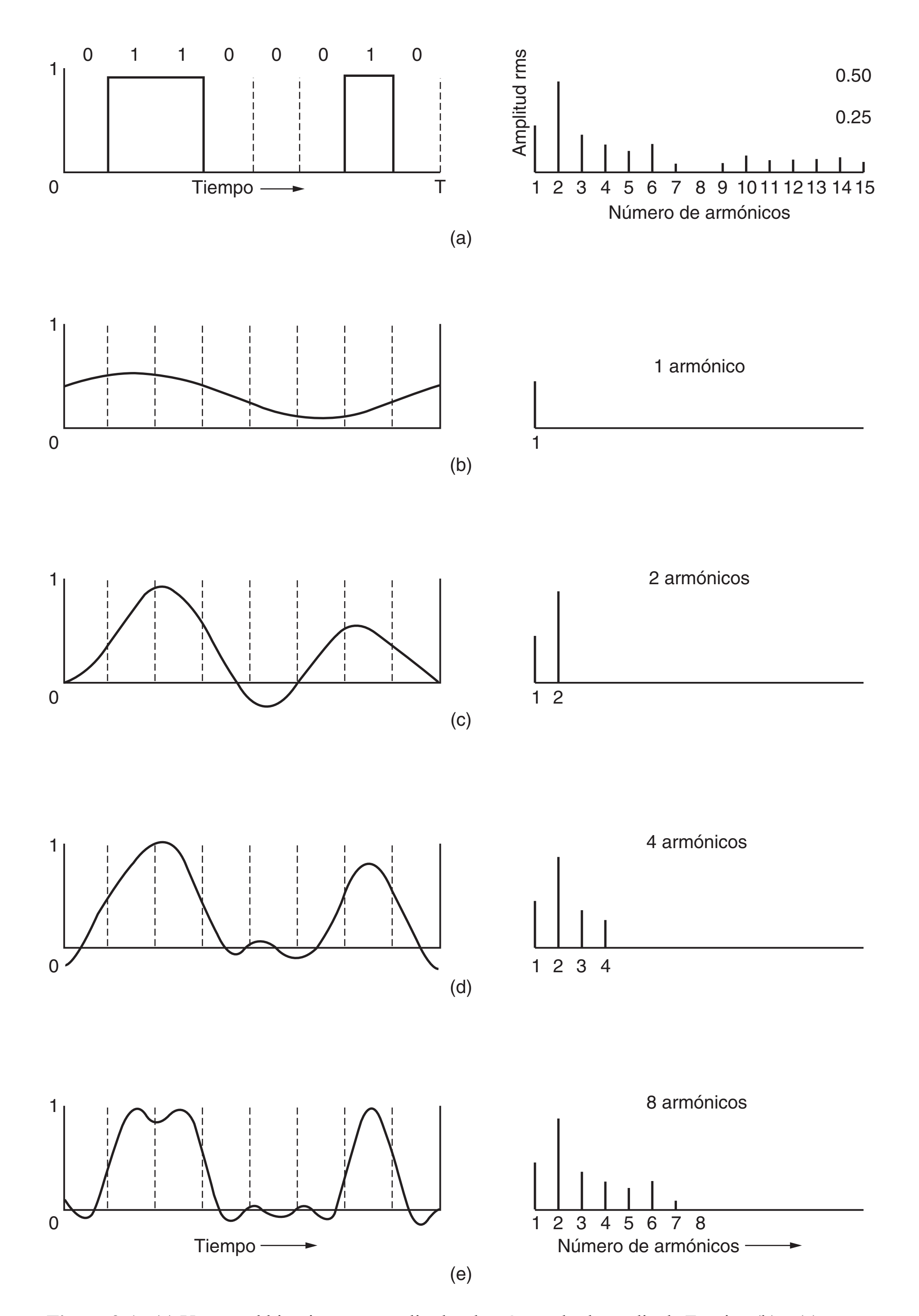


Figura 2-1. (a) Una señal binaria y sus amplitudes de raíz cuadrada media de Fourier. (b) a (e) Aproximaciones sucesivas a la señal original.

Bps	T (mseg)	Primer armónico (Hz)	Núm. de armónico transmitidos
300	26.67	37.5	80
600	13.33	75	40
1200	6.67	150	20
2400	3.33	300	10
4800	1.67	600	5
9600	0.83	1200	2
19200	0.42	2400	1
38400	0.21	4800	0

Figura 2-2. Relación entre la tasa de datos y los armónicos para nuestro ejemplo.

Hay mucha confusión en cuanto al ancho de banda, ya que tiene distintos significados para los ingenieros eléctricos y para los científicos de computadoras. Para los ingenieros eléctricos, el ancho de banda (analógico) es (como lo describimos antes) una cantidad que se mide en Hz. Para los científicos de computadora, el ancho de banda (digital) es la tasa de datos máxima de un canal, una cantidad que se mide en bits/segundo. Esa tasa de datos es el resultado final de usar el ancho de banda analógico de un canal físico para transmisión digital, y ambos están relacionados, como veremos a continuación. En este libro dejaremos en claro mediante el contexto si nos referimos al ancho de banda analógico (Hz) o al digital (bits/seg).

2.1.3 La tasa de datos máxima de un canal

En 1924, un ingeniero de AT&T llamado Henry Nyquist se dio cuenta de que incluso un canal perfecto tiene una capacidad de transmisión finita y dedujo una ecuación para expresar la tasa de datos máxima para un canal sin ruido con un ancho de banda finito. En 1948, Claude Shannon retomó el trabajo de Nyquist y lo extendió al caso de un canal sujeto a ruido aleatorio (es decir, termodinámico) (Shannon, 1948). Este documento es el más importante en toda la teoría de la información. Aquí sólo resumiremos brevemente sus resultados, que ahora son clásicos.

Nyquist demostró que si se pasa una señal cualquiera a través de un filtro pasa-bajas con un ancho de banda B, la señal filtrada se puede reconstruir por completo tomando sólo $2B$ muestras (exactas) por segundo. No tiene caso muestrear la línea más de $2B$ veces por segundo, ya que los componentes de mayor frecuencia que dicho muestreo pudiera recuperar ya se han filtrado. Si la señal consiste en V niveles discretos, el teorema de Nyquist establece lo siguiente:

$$\text{Tasa de datos máxima} = 2B \log_2 V \text{ bits/seg} \qquad (2\text{-}2)$$

Por ejemplo, un canal sin ruido de 3kHz no puede transmitir señales binarias (de dos niveles) a una velocidad mayor de 6000 bps.

Hasta ahora hemos considerado sólo los canales sin ruido. Si hay ruido aleatorio presente, la situación se deteriora con rapidez. Y siempre hay ruido aleatorio (térmico) presente debido al movimiento de las moléculas en el sistema. La cantidad de ruido térmico presente se mide con base en la relación entre la potencia de la señal y la potencia del ruido; llamada **SNR** (**Relación Señal a Ruido**, del inglés *Signal-to-Noise Ratio*). Si denotamos la potencia de la señal mediante S y la potencia del ruido mediante N, la relación señal a ruido es S/N. Por lo general la relación se expresa en una escala de logaritmo como la cantidad $10 \log_{10}$

S/*N*, ya que puede variar sobre un gran rango. Las unidades de esta escala logarítmica se llaman **decibeles** (**dB**), en donde "deci" significa 10 y "bel" se eligió en honor a Alexander Graham Bell, inventor del teléfono. Una relación *S*/*N* de 10 es igual a 10 dB, una relación de 100 es igual a 20 dB, una relación de 1000 es igual a 30 dB y así sucesivamente. A menudo los fabricantes de amplificadores estereofónicos caracterizan el ancho de banda (rango de frecuencia) en el cual sus productos son lineales dando la frecuencia de 3 dB en cada extremo. Éstos son los puntos en los que el factor de amplificación se ha dividido de manera aproximada a la mitad (puesto que $10 \log_{10} 0.5 \approx -3$).

El principal resultado de Shannon es que la tasa de datos máxima (o **capacidad**) de un canal ruidoso, cuyo ancho de banda es *B* Hz y cuya relación señal a ruido es *S*/*N*, está dada por:

$$\text{Número máximo de bits/seg} = B \log_2 (1 + S/N) \qquad (2\text{-}3)$$

Esto nos indica las mejores capacidades que pueden tener los canales reales. Por ejemplo, la **ADSL** (**Línea Asimétrica de Suscriptor Digital**) que provee acceso a Internet a través de líneas telefónicas comunes, utiliza un ancho de banda de aproximadamente 1 MHz. La SNR depende en gran parte de la distancia entre el hogar y la central telefónica; una SNR de alrededor de 40 dB para líneas cortas de 1 a 2 km es algo muy bueno. Con estas características, el canal nunca podrá transmitir a más de 13 Mbps, sin importar cuántos niveles de señal se utilicen ni con qué frecuencia se tomen las muestras. En la práctica, el servicio ADSL se especifica hasta 12 Mbps, aunque es frecuente que los usuarios vean tasas más bajas. En realidad esta tasa de datos es muy buena, con más de 60 años de técnicas de comunicaciones que han reducido la brecha entre la capacidad de Shannon y la capacidad de los sistemas reales.

El resultado de Shannon se derivó de los argumentos de la teoría de la información y se aplica a cualquier canal sujeto a ruido térmico. Los ejemplos contrarios se deben clasificar en la misma categoría que las máquinas de movimiento perpetuo. Para que el servicio ADSL sobrepase los 13 Mbps, debe mejorar la SNR (por ejemplo, insertando repetidores digitales en las líneas más cercanas a los clientes) o usar más ancho de banda, como se está haciendo con la evolución al servicio ADSL2+.

2.2 MEDIOS DE TRANSMISIÓN GUIADOS

El propósito de la capa física es transportar bits de una máquina a otra. Se pueden utilizar varios medios físicos para la transmisión real. Cada medio tiene su propio nicho en términos de ancho de banda, retardo, costo y facilidad de instalación y mantenimiento. A grandes rasgos, los medios se agrupan en medios guiados (como el cable de cobre y la fibra óptica) y en medios no guiados (como la transmisión inalámbrica terrestre, los satélites y los láseres a través del aire). En esta sección veremos los medios guiados y en la siguiente los medios no guiados.

2.2.1 Medios magnéticos

Una de las formas más comunes para transportar datos de una computadora a otra es almacenarlos en cinta magnética o medios removibles (por ejemplo, DVD regrabables), transportar físicamente la cinta o los discos a la máquina de destino y leerlos de nuevo. Aunque este método no es tan sofisticado como usar un satélite de comunicación geosíncrono, a menudo es mucho más rentable, en especial para las aplicaciones en las que el ancho de banda alto o el costo por bit transportado es el factor clave.

Para aclarar este punto veremos un simple cálculo. Una cinta Ultrium estándar puede guardar 800 gigabytes. Una caja de $60 \times 60 \times 60$ cm puede contener aproximadamente 1000 de estas cintas, para una capacidad total de 800 terabytes, o 6400 terabits (6.4 petabits). Una caja de cintas se puede entregar en

cualquier parte de Estados Unidos en un plazo de 24 horas a través de Federal Express o cualquier otra empresa de mensajería. El ancho de banda efectivo de esta transmisión es de 6400 terabits/86,400 seg, o de un bit sobre 70 Gbps. Si el destino sólo está a una hora de camino por carretera, el ancho de banda se incrementa a cerca de 1700 Gbps. Ninguna red de computadoras puede siquiera acercarse a esta velocidad. Claro que las redes se están haciendo más veloces, pero las densidades de las cintas también están aumentando.

Si ahora analizamos el costo, obtenemos una imagen similar. El costo aproximado de una cinta Ultrium es de $40 si se compra al mayoreo.* Una cinta se puede reutilizar por lo menos 10 veces, de modo que el costo de la cinta puede ser de $4000 por caja, por cada uso. Si agregamos $1000 de envío (probablemente sea mucho menos), el costo aproximado de enviar 800 Tb es de $5000. Esto representa enviar un gigabyte por un poco más de medio centavo. Ninguna red puede vencer eso. La moraleja de la historia es:

Nunca subestime el ancho de banda de una camioneta repleta de cintas que viaje a toda velocidad por la carretera.

2.2.2 Par trenzado

Aunque las características de ancho de banda de la cinta magnética son excelentes, las características de retardo son pobres. El tiempo de transmisión se mide en minutos u horas, no en milisegundos. Para muchas aplicaciones se necesita una conexión en línea. Uno de los medios de transmisión más antiguos y todavía el más común es el **par trenzado**. Un par trenzado consta de dos cables de cobre aislados, por lo general de 1 mm de grosor. Los cables están trenzados en forma helicoidal, justo igual que una molécula de ADN. El trenzado se debe a que dos cables paralelos constituyen una antena simple. Cuando se trenzan los cables, las ondas de distintos trenzados se cancelan y el cable irradia con menos efectividad. Por lo general una señal se transmite como la diferencia en el voltaje entre los dos cables en el par. Esto ofrece una mejor inmunidad al ruido externo, ya que éste tiende a afectar ambos cables en la misma proporción y en consecuencia, el diferencial queda sin modificación.

La aplicación más común del par trenzado es el sistema telefónico. Casi todos los teléfonos se conectan a la central telefónica mediante un par trenzado. Tanto las llamadas telefónicas como el acceso ADSL a Internet se llevan a cabo mediante estas líneas. Se pueden tender varios kilómetros de par trenzado sin necesidad de amplificación, pero en distancias mayores la señal se atenúa demasiado y se requieren repetidores. Cuando muchos pares trenzados se tienden en paralelo a una distancia considerable, como los cables que van de un edificio de apartamentos a la central telefónica, se agrupan en un haz y se cubren con una funda protectora. Los pares en estos haces interferirían unos con otros si no estuvieran trenzados. En algunas partes del mundo en donde las líneas telefónicas penden de postes sobre la tierra, es común ver haces de varios centímetros de diámetro.

Los pares trenzados se pueden usar para transmitir la información analógica o digital. El ancho de banda depende del grosor del cable y de la distancia que recorre, pero en muchos casos se pueden lograr varios megabits/seg durante pocos kilómetros. Debido a su adecuado desempeño y bajo costo, los pares trenzados se utilizan mucho y es probable que se sigan utilizando durante varios años más.

Existen diversos tipos de cableado de par trenzado. El que se utiliza con mayor frecuencia en muchos edificios de oficinas se llama cable de **categoría 5**, o “cat 5”. Un par trenzado de categoría 5 consta de dos cables aislados que se trenzan de manera delicada. Por lo general se agrupan cuatro de esos pares en una funda de plástico para protegerlos y mantenerlos juntos. Este arreglo se muestra en la figura 2-3.

* Las cantidades indicadas están en dólares estadounidenses, y son sólo ilustrativas. (N. del E.)

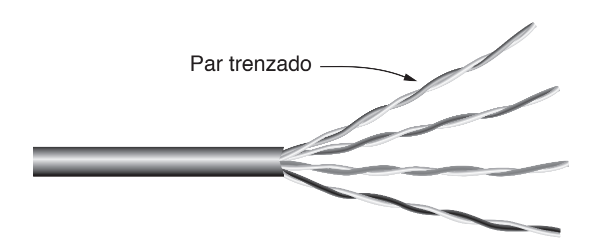


Figura 2-3. Cable UTP categoría 5 con cuatro pares trenzados.

Los distintos estándares de LAN pueden utilizar los pares trenzados de manera diferente. Por ejemplo, el estándar Ethernet de 100 Mbps utiliza dos (de los cuatro) pares, uno para cada dirección. Para llegar a velocidades más altas, el estándar Ethernet de 1 Gbps utiliza los cuatro pares en ambas direcciones al mismo tiempo; para ello el receptor debe eliminar la señal que se transmite en forma local.

Ahora es buen momento para ver cierta terminología general. Los enlaces que se pueden utilizar en ambas direcciones al mismo tiempo, como un camino con dos carriles, se llaman enlaces **full-dúplex**. En contraste, los enlaces que se pueden utilizar en cualquier dirección, pero sólo una a la vez, como una vía de ferrocarril de un solo sentido, se llaman enlaces **half-dúplex**. Hay una tercera categoría que consiste en enlaces que permiten tráfico sólo en una dirección, como una calle de un solo sentido. A éstos se les conoce como enlaces **simplex**.

Ahora volvamos al par trenzado. Los cables de categoría 5 reemplazaron a los cables de **categoría 3** con un cable similar que utiliza el mismo conector, pero tiene más trenzas por metro. Entre más trenzas, hay menos diafonía y se logra una señal de mejor calidad a distancias más largas, lo que hace a los cables más adecuados para la comunicación de computadoras de alta velocidad, en especial para las redes LAN de 100 Mbps y de 1 Gbps.

Es muy probable que el nuevo cableado sea de **categoría 6** o incluso de **categoría 7**. Estas categorías tienen especificaciones más estrictas para manejar señales con mayores anchos de banda. Algunos cables de categoría 6 y superiores están estimados para señales de 500 MHz y pueden soportar los enlaces de 10 Gbps que se implementarán en un futuro cercano.

A los tipos de cables hasta la categoría 6 se les conoce como **UTP** (**Par Trenzado sin Blindaje**, del inglés *Unshielded Twisted Pair*), ya que están constituidos tan sólo de alambres y aislantes. En contraste, los cables de categoría 7 tienen blindaje en cada uno de los pares trenzados por separado, así como alrededor de todo el cable (pero dentro de la funda protectora de plástico). El blindaje reduce la susceptibilidad a interferencias externas y la diafonía con otros cables cercanos para cumplir con las especificaciones más exigentes de rendimiento. Estos cables son parecidos a los de par trenzado con blindaje de alta calidad pero voluminosos y costosos, que IBM introdujo a principios de la década de 1980, pero que nunca fueron populares fuera de las instalaciones de IBM. Sin duda, ahora es tiempo de intentarlo de nuevo.

2.2.3 Cable coaxial

El **cable coaxial** es otro medio de transmisión común (conocido simplemente como "coax"). Este cable tiene mejor blindaje y mayor ancho de banda que los pares trenzados sin blindaje, por lo que puede abarcar mayores distancias a velocidades más altas. Hay dos tipos de cable coaxial que se utilizan ampliamente. El de 50 ohms es uno de ellos y se utiliza por lo general cuando se tiene pensado emplear una transmisión digital desde el inicio. El otro tipo es el de 75 ohms y se utiliza para la transmisión analógica y la televisión por cable. Esta distinción se basa en factores históricos más que técnicos (por ejemplo, las primeras antenas de dipolos tenían una impedancia de 300 ohms y era fácil usar los transformadores adaptadores

de impedancia de 4:1). A partir de la década de 1990, los operadores de TV por cable empezaron a proveer acceso a Internet por cable, de modo que el cable de 75 ohms se ha vuelto más importante para la comunicación de datos.

Un cable coaxial consiste en alambre de cobre rígido como núcleo, rodeado por un material aislante. El aislante está forrado de un conductor cilíndrico, que por lo general es una malla de tejido fuertemente trenzado. El conductor externo está cubierto con una funda protectora de plástico. En la figura 2-4 se muestra una vista seccionada de un cable coaxial.

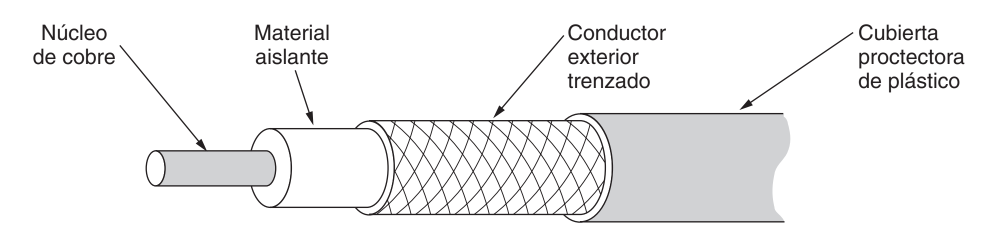


Figura 2-4. Un cable coaxial.

Gracias a su construcción y blindaje, el cable coaxial tiene una buena combinación de un alto ancho de banda y una excelente inmunidad al ruido. El ancho de banda posible depende de la calidad y la longitud del cable. Los cables modernos tienen un ancho de banda de hasta unos cuantos GHz. Los cables coaxiales solían utilizarse mucho dentro del sistema telefónico para las líneas de larga distancia, pero ahora se reemplazaron en su mayoría por fibra óptica en las rutas de largo recorrido. Sin embargo, el cable coaxial se sigue utilizando mucho para la televisión por cable y las redes de área metropolitana.

2.2.4 Líneas eléctricas

Las redes de telefonía y de televisión por cable no son las únicas fuentes de cableado que se pueden reutilizar para la comunicación de datos. Hay otro tipo más común de cableado: las líneas de energía eléctrica. Estas líneas transportan energía eléctrica a las casas, y el cableado eléctrico dentro de las casas distribuye la energía a las tomas de corriente.

El uso de las líneas eléctricas para la comunicación de datos es una idea antigua. Las compañías de electricidad han utilizado las líneas eléctricas para la comunicación de baja velocidad durante varios años (por ejemplo, la medición remota), así como también en el hogar para controlar dispositivos (por ejemplo, el estándar X10). En años recientes surgió un interés renovado por la comunicación de alta velocidad a través de estas líneas, tanto dentro del hogar con una LAN como fuera de éste para el acceso a Internet de banda ancha. Nos concentraremos en el escenario más común: el uso de los cables eléctricos dentro del hogar.

La conveniencia de usar líneas eléctricas para el trabajo en red debe quedar claro. Simplemente se debe conectar una TV y un receptor a la toma de pared, lo cual debe hacer de todas formas, ya que necesitan electricidad para que puedan enviar y recibir películas a través del cableado eléctrico. En la figura 2-5 se muestra esta configuración. No hay otro clavija o radio transmisor. La señal de datos está sobrepuesta en la señal eléctrica de baja frecuencia (en el cable activo o "caliente"), ya que ambas señales usan el cableado al mismo tiempo.

La dificultad al utilizar el cableado eléctrico en el hogar para una red es que éste se diseñó para distribuir señales eléctricas. Esta tarea es muy distinta a la de distribuir señales de datos, en donde el cableado

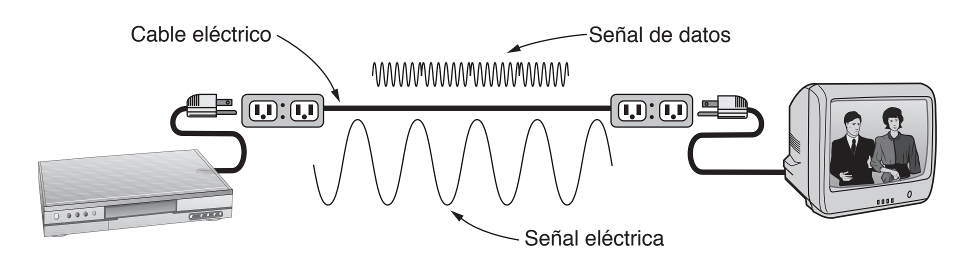


Figura 2-5. Una red que utiliza el cableado eléctrico en el hogar.

del hogar hace un mal trabajo. Las señales eléctricas se envían a 50-60 Hz y el cableado atenúa las señales de frecuencia mucho más altas (MHz) que se necesitan para la comunicación de datos de alta velocidad. Las propiedades eléctricas del cableado varían de una casa a la otra y cambian a medida que los electrodomésticos se encienden y apagan, lo cual hace que las señales de datos reboten alrededor del cableado. Las corrientes transitorias que se producen al encender y apagar los electrodomésticos crean ruido eléctrico a través de un amplio rango de frecuencias. Y sin el cuidadoso trenzado de los pares trenzados, el cableado eléctrico actúa como una antena simple que recoge las señales externas e irradia sus propias señales. Este comportamiento significa que para cumplir con los requerimientos regulatorios, la señal de datos debe excluir las frecuencias que se otorgan bajo licencia, como las bandas de radioaficionados.

A pesar de estas dificultades, es práctico enviar por lo menos 100 Mbps a través del cableado eléctrico en el hogar mediante el uso de esquemas de comunicación que resisten las frecuencias dañadas y las ráfagas de errores. Muchos productos utilizan varios estándares propietarios para las redes de la línea eléctrica, así que los estándares internacionales se encuentran activamente en desarrollo.

2.2.5 Fibra óptica

Muchas personas en la industria de la computación sienten un enorme orgullo por la rapidez con que la tecnología de las computadoras está mejorando según la ley de Moore, la cual predice una duplicación de la cantidad de transistores por chip aproximadamente cada dos años (Schaller, 1997). La PC original de IBM (1981) operaba a una velocidad de reloj de 4.77 MHz. Veintiocho años después, las PC pueden operar una CPU de cuatro núcleos a 3 GHz. Este incremento es una ganancia de un factor aproximado de 2 500, o de 16 por década. Impresionante.

En el mismo periodo, los enlaces de comunicación de área amplia pasaron de 45 Mbps (una línea T3 en el sistema telefónico) a 100 Gbps (una línea moderna de larga distancia). Esta ganancia es igual de impresionante, mayor a un factor de 2 000 y cerca de 16 por década, mientras que al mismo tiempo la tasa de error se redujo de 10^{-5} por bit a casi cero. Asimismo, los CPU individuales están empezando a llegar a los límites físicos, razón por la cual ahora se incrementa el número de CPU por chip. En contraste, el ancho de banda que se puede lograr con la tecnología de fibra óptica es mayor a 50 000 Gbps (50 Tbps) y no estamos siquiera cerca de llegar a esos límites. El límite práctico actual de cerca de 100 Gbps se debe a nuestra incapacidad de realizar conversiones entre las señales eléctricas y ópticas con más rapidez. Para construir enlaces de mayor capacidad, simplemente se transportan muchos canales en paralelo a través de una sola fibra óptica.

En esta sección estudiaremos la fibra óptica para aprender cómo funciona esta tecnología de transmisión. En la carrera continua entre la computación y la comunicación, tal vez gane la comunicación debido a las redes de fibra óptica. La implicación de esto sería en esencia un ancho de banda infinito y un nuevo acuerdo convencional para establecer que las computadoras son irremediablemente lentas, de modo que las redes deben tratar de evitar las tareas de cómputo a toda costa, sin importar qué tanto ancho de banda

se desperdicie. Este cambio tardará un tiempo en penetrar en una generación de científicos e ingenieros en computación acostumbrados a pensar en términos de los bajos límites de Shannon impuestos por el cobre.

Desde luego que este escenario no describe toda la historia, ya que no incluye el enorme costo para instalar fibra óptica en la última milla para llegar a los clientes y evitar el bajo ancho de banda de los cables y la disponibilidad limitada del espectro. Además, se requiere más energía para mover bits que para realizar cálculos. Tal vez siempre tengamos islas de inequidades en donde la computación o la comunicación sean esencialmente gratuitas. Por ejemplo, al margen de Internet maximizamos el uso de la computación y el almacenamiento para combatir el problema de comprimir el contenido y usar cachés, todo para mejorar el uso de los vínculos de acceso a Internet. Dentro de Internet tal vez hagamos lo contrario, con compañías como Google que desplazan grandes cantidades de datos por la red hacia donde sea más económico almacenarlos o realizar cálculos con ellos.

La fibra óptica se utiliza para la transmisión de larga distancia en las redes troncales, las redes LAN de alta velocidad (aunque hasta ahora el cobre siempre ha logrado ponerse a la par) y el acceso a Internet de alta velocidad como **FTTH** (**Fibra para el Hogar**, del inglés *Fiber To The Home*). Un sistema de transmisión óptico tiene tres componentes clave: la fuente de luz, el medio de transmisión y el detector. Por convención, un pulso de luz indica un bit 1 y la ausencia de luz indica un bit 0. El medio de transmisión es una fibra de vidrio ultradelgada. El detector genera un pulso eléctrico cuando la luz incide en él. Al conectar una fuente de luz a un extremo de una fibra óptica y un detector al otro extremo, tenemos un sistema de transmisión de datos unidireccional que acepta una señal eléctrica, la convierte y la transmite mediante pulsos de luz, y después reconvierte la salida a una señal eléctrica en el extremo receptor.

Este sistema de transmisión tendría fugas de luz y sería inútil en la práctica si no fuera por un interesante principio de la física. Cuando un rayo de luz pasa de un medio a otro (por ejemplo, de sílice fundida al aire), el rayo se refracta (dobla) en el límite la sílice y el aire, como se muestra en la figura 2-6(a). Aquí vemos un rayo de luz que incide en el límite a un ángulo α_1 que emerge con un ángulo β_1. El grado de refracción depende de las propiedades de los dos medios (en especial, de sus índices de refracción). Para ángulos de incidencia por encima de cierto valor crítico, la luz se refracta de regreso a la sílice; nada de ella escapa al aire. Por ende, un rayo de luz incidente con un ángulo igual o mayor al crítico que queda atrapado dentro de la fibra, como se muestra en la figura 2-6(b), y se puede propagar por muchos kilómetros prácticamente sin pérdidas.

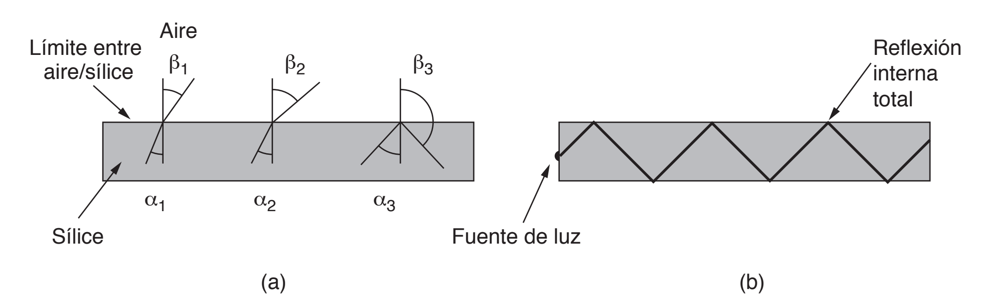


Figura 2-6. (a) Tres ejemplos de un rayo de luz desde el interior de una fibra de sílice que incide sobre el límite entre aire y sílice a distintos ángulos. (b) Luz atrapada por reflexión interna total.

El bosquejo de la figura 2-6(b) sólo muestra un rayo atrapado, pero como cualquier rayo de luz que incida en la frontera por encima del ángulo crítico se reflejará de manera interna, habrá muchos rayos distintos rebotando con ángulos diferentes. Se dice que cada rayo tiene un modo distinto, por lo que una fibra con esta propiedad se llama **fibra multimodal**.

Pero si el diámetro de la fibra se reduce a unas cuantas longitudes de onda de luz, la fibra actúa como una guía de ondas y la luz se puede propagar sólo en línea recta, sin rebotar, con lo que se obtiene una **fibra monomodo**. Estas fibras son más costosas pero se utilizan mucho para distancias más largas. Las fibras monomodo disponibles en la actualidad pueden transmitir datos a 100 Gbps por 100 km sin necesidad de amplificación. Incluso se han logrado tasas de datos más altas en el laboratorio, para distancias más cortas.

Transmisión de luz a través de fibras

Las fibras ópticas están hechas de vidrio, que a su vez se fabrica a partir de la arena, una materia prima de bajo costo disponible en cantidades ilimitadas. La fabricación del vidrio era conocida por los antiguos egipcios, pero su vidrio no podía ser mayor de 1 mm de grosor para que la luz pudiera atravesarlo. Durante el Renacimiento se desarrolló un vidrio lo bastante transparente como para usarlo en las ventanas. El vidrio utilizado para las fibras ópticas modernas es tan transparente que si los océanos estuvieran llenos de él en vez de agua, el lecho marino sería tan visible desde la superficie como lo es el suelo desde un avión en un día claro.

La atenuación de la luz que pasa por el vidrio depende de la longitud de onda de la luz (así como de algunas propiedades físicas del vidrio). Se define como la relación entre la potencia de la señal de entrada y la de salida. Para el tipo de vidrio que se utiliza en las fibras ópticas, la atenuación se muestra en la figura 2-7 en unidades de decibeles por kilómetro lineal de fibra. Por ejemplo, un factor de pérdida de potencia de la señal de dos nos da una atenuación de $10 \log_{10} 2 = 3$ dB. La figura muestra la parte cercana al infrarrojo del espectro, que es lo que se utiliza en la práctica. La luz visible tiene longitudes de onda ligeramente más cortas, de 0.4 a 0.7 micras (1 micra equivale a 10^{-6} metros). El verdadero purista métrico se referiría a estas longitudes como de 400 nm a 700 nm, pero nos apegaremos al uso tradicional.

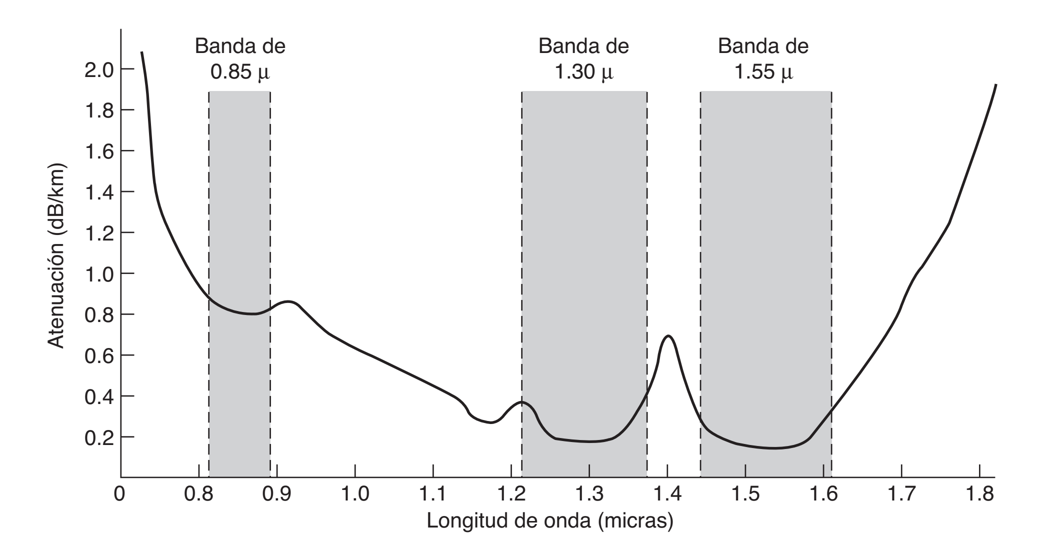


Figura 2-7. Atenuación de la luz dentro de una fibra en la región de infrarrojo.

En la actualidad se utilizan mucho tres bandas de longitud de onda para la comunicación óptica. Estas tres bandas se centran en 0.85, 1.30 y 1.55 micras, respectivamente. Las tres bandas tienen de 25 000 a 30 000 GHz de amplitud. La banda de 0.85 micras se utilizó primero. Tiene una mayor atenuación y, por lo tanto, se utiliza para distancias más cortas, pero a esa longitud de onda se pueden fabricar láseres y componentes electrónicos con el mismo material (arseniuro de galio). Las últimas dos bandas tienen buenas propiedades de atenua-

ción (una pérdida de menos de 5% por cada kilómetro). Hoy en día, la banda de 1.55 micrones se utiliza mucho en los amplificadores dopados con erbio que trabajan directamente en el domino óptico.

La longitud de los pulsos de luz que se transmiten por una fibra aumenta conforme se propagan. A este fenómeno se le conoce como **dispersión cromática**. Su magnitud depende de la longitud de onda. Una forma de evitar que se traslapen estos pulsos dispersos es aumentar la distancia entre ellos, pero esto se puede hacer sólo si se reduce la tasa de transmisión. Por fortuna se descubrió que si se da a los pulsos una forma especial relacionada con el recíproco del coseno hiperbólico, se cancelan casi todos los efectos de la dispersión y es posible enviar pulsos a miles de kilómetros sin una distorsión apreciable de la forma. Estos pulsos se llaman **solitones**. Se está realizando una cantidad considerable de investigaciones para sacar los solitones del laboratorio y llevarlos al campo.

Cables de fibras

Los cables de fibra óptica son similares a los coaxiales, excepto por el trenzado. En la figura 2-8(a) aparece una fibra óptica individual, vista de lado. Al centro se encuentra el núcleo de vidrio, a través del cual se propaga la luz. En las fibras multimodales, el núcleo es por lo general de 50 micras de diámetro, aproximadamente el grosor de un cabello humano. En las fibras de monomodo, el núcleo es de 8 a 10 micras.

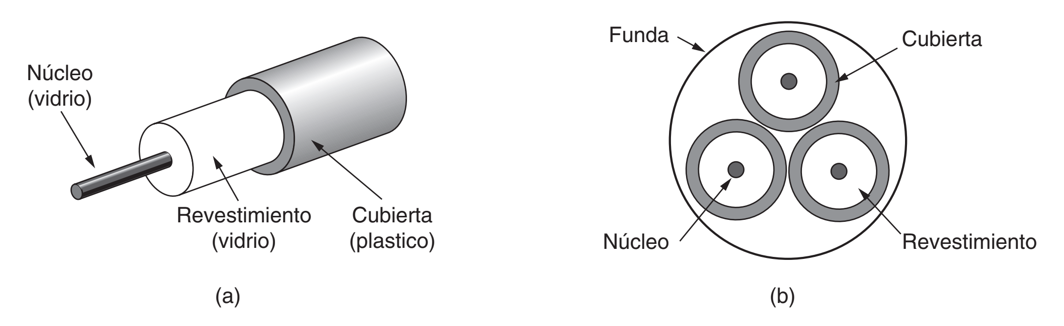


Figura 2-8. (a) Vista lateral de una sola fibra. (b) Vista de extremo de una envoltura con tres fibras.

El núcleo está rodeado de un revestimiento de vidrio con un índice de refracción más bajo que el del núcleo, con el fin de mantener toda la luz en el núcleo. Después viene una cubierta delgada de plástico para proteger el revestimiento. Por lo general las fibras se agrupan en haces, protegidas por una funda exterior. La figura 2-8(b) muestra una funda con tres fibras.

Por lo general, las fundas de fibras terrestres se colocan un metro debajo de la superficie, en donde en ocasiones están sujetas a los ataques de retroexcavadoras o topos. Cerca de la costa, las fundas de fibras transoceánicas se entierran en zanjas mediante una especie de arado marino. En aguas profundas, simplemente se colocan en el fondo, donde pueden ser enganchadas por pesqueros o atacadas por un calamar gigante.

Las fibras se pueden conectar de tres maneras distintas. Primera, pueden terminar en conectores e insertarse en clavijas de fibra. Los conectores pierden entre un 10 y 20% de la luz, pero facilitan la reconfiguración de los sistemas.

Segunda, se pueden empalmar en forma mecánica. Los empalmes mecánicos simplemente acomodan los dos extremos cortados con cuidado, uno junto a otro en una manga especial y los sujetan en su lugar. Para mejorar la alineación se puede pasar luz a través de la unión para después realizar pequeños ajustes de modo que se maximice la señal. El personal capacitado tarda cerca de cinco minutos en crear empalmes mecánicos y se produce una pérdida de luz de 10%.

Tercera, se pueden fusionar (fundir) dos piezas de fibra para formar una conexión sólida. Un empalme por fusión es casi tan bueno como una sola fibra, pero incluso en este caso se produce una pequeña cantidad de atenuación.

En los tres tipos de empalmes se pueden producir reflejos en el punto del empalme; además la energía reflejada puede interferir con la señal.

Por lo general se utilizan dos tipos de fuentes de luz para producir las señales: **LED** (**Diodos Emisores de Luz**, del inglés *Light Emitting Diodes*) y láseres semiconductores. Estas fuentes de luz tienen distintas propiedades, como se muestra en la figura 2-9. Se pueden optimizar en cuanto a la longitud de onda, para lo cual se insertan interferómetros Fabry-Perot o Mach-Zehnder entre la fuente y la fibra. Los interferómetros Fabry-Perot son simples cavidades resonantes que consisten en dos espejos paralelos. La luz incide en los espejos en forma perpendicular. La longitud de la cavidad separa las longitudes de onda que caben dentro de un número entero de veces. Los interferómetros Mach-Zehnder separan la luz en dos haces, los cuales viajan distancias ligeramente distintas. Se vuelven a combinar en el extremo y están en fase sólo para ciertas longitudes de onda.

Característica	LED	Láser semiconductor
Tasa de datos	Baja	Alta
Tipo de fibra	Multimodo	Multimodo o monomodo
Distancia	Corta	Larga
Tiempo de vida	Vida larga	Vida corta
Sensibilidad a la temperatura	Poca	Considerable
Costo	Bajo	Elevado

Figura 2-9. Comparación de los diodos semiconductores y los LED como fuentes de luz.

El extremo receptor de una fibra óptica consiste en un fotodiodo, el cual emite un pulso eléctrico cuando lo golpea la luz. El tiempo de respuesta de los fotodiodos, que convierten la señal óptica en eléctrica, limita la tasa de datos a cerca de 100 Gbps. El ruido térmico es otro inconveniente, por lo que un pulso de luz debe llevar suficiente potencia para detectarlo. Cuando los pulsos tienen la potencia suficiente, la tasa de error se puede reducir de manera considerable.

Comparación entre la fibra óptica y el alambre de cobre

Es ilustrativo comparar la fibra con el cobre. La fibra tiene muchas ventajas. Para empezar, puede manejar anchos de banda mucho mayores que el cobre. Tan sólo por esto sería indispensable en las redes de alto rendimiento. Debido a la baja atenuación, sólo se necesitan repetidores aproximadamente cada 50 km en líneas extensas, mientras que el cobre requiere repetidores cada 5 km, lo cual implica un ahorro considerable en el costo. La fibra también tiene la ventaja de que no le afectan las sobrecargas de energía, la interferencia electromagnética ni los cortes en el suministro de energía. Tampoco le afectan las sustancias corrosivas en el aire, lo cual es importante en los ambientes industriales pesados.

Por extraño que parezca, a las compañías telefónicas les gusta la fibra por una razón distinta: es delgada y ligera. Muchos conductos de cables existentes están llenos por completo, de modo que no hay espacio para agregar más capacidad. Si se quita todo el cobre y se sustituye por fibra se vacían los conductos, además de que el cobre tiene un excelente valor de reventa para las refinerías de cobre, quienes lo ven como materia prima de alta calidad. Asimismo, la fibra es mucho más ligera que el cobre. Mil pares

trenzados de 1 km de longitud pesan 8 000 kg. Dos fibras tienen más capacidad y sólo pesan 100 kg, lo cual reduce la necesidad de costosos sistemas mecánicos de apoyo a los que se debe dar mantenimiento. Para las nuevas rutas, la fibra es la mejor opción debido a que su costo de instalación es mucho más bajo. Por último, las fibras no tienen fugas de luz y son difíciles de intervenir. Estas propiedades les confieren una excelente seguridad contra los potenciales espías.

Sin embargo, la fibra es una tecnología poco familiar que requiere habilidades que no todos los ingenieros tienen, además se pueden dañar con facilidad si se les dobla demasiado. Como la transmisión óptica es unidireccional por naturaleza, para la comunicación en ambos sentidos se requieren ya sea dos fibras o dos bandas de frecuencia en una fibra. Por último, las interfaces de las fibras cuestan más que las interfaces eléctricas. Sin embargo, el futuro de todas las comunicaciones fijas de datos a distancias, de algo más que unos cuantos metros, en definitiva está en la fibra. Para un análisis detallado de todos los aspectos de la fibra óptica y sus redes, consulte a Hecht (2005).

2.3 TRANSMISIÓN INALÁMBRICA

Nuestra Era ha dado origen a los adictos a la información: personas que necesitan estar todo el tiempo en línea. Para estos usuarios móviles no son de utilidad el par trenzado, el cable coaxial ni la fibra óptica. Necesitan obtener datos para sus computadoras laptop, notebook, de bolsillo, de mano o de reloj de pulsera sin tener que estar atados a la infraestructura de comunicación terrestre. Para estos usuarios, la comunicación inalámbrica es la respuesta.

En las siguientes secciones analizaremos la comunicación inalámbrica en general, la cual tiene muchas otras aplicaciones importantes además de proveer conectividad a los usuarios que desean navegar en la Web desde la playa. La tecnología inalámbrica ofrece ventajas incluso para dispositivos fijos en ciertos casos. Por ejemplo, si es difícil tender una fibra hasta un edificio debido al terreno (montañas, junglas, pantanos, etc.), tal vez sea mejor usar tecnología inalámbrica. Cabe mencionar que la comunicación digital inalámbrica moderna se inició en las islas de Hawai, en donde largos tramos del Océano Pacífico separaban a los usuarios de su centro de cómputo y el sistema telefónico era inadecuado.

2.3.1 El espectro electromagnético

Cuando los electrones se mueven, crean ondas electromagnéticas que se pueden propagar por el espacio (incluso en el vacío). El físico inglés James Clerk Maxwell predijo estas ondas en 1865 y el físico alemán Heinrich Hertz las observó por primera vez en 1887. El número de oscilaciones por segundo de una onda es su **frecuencia**, f, y se mide en **Hz** (en honor de Heinrich Hertz). La distancia entre dos máximos (o mínimos) consecutivos se llama **longitud de onda** y se designa en forma universal mediante la letra griega λ (lambda).

Al conectar una antena del tamaño apropiado a un circuito eléctrico, las ondas electromagnéticas se pueden difundir de manera eficiente y un receptor las puede captar a cierta distancia. Toda la comunicación inalámbrica se basa en este principio.

En el vacío, todas las ondas electromagnéticas viajan a la misma velocidad sin importar cuál sea su frecuencia. Esta velocidad se conoce como **velocidad de la luz**, c, y es de aproximadamente 3×10^8 m/seg, o alrededor de 1 pie (30 cm) por nanosegundo (podríamos argumentar para redefinir el pie como la distancia que viaja la luz en un vacío por 1 nseg en vez de basarnos en el tamaño del zapato de un rey que murió hace mucho tiempo). En el cobre o la fibra, la velocidad baja a casi 2/3 de este valor y se vuelve ligeramente dependiente de la frecuencia. La velocidad de la luz es el máximo límite de velocidad. Ningún objeto o señal puede llegar a ser más rápido que la luz.

La relación fundamental entre f, λ y c (en el vacío) es

$$\lambda f = c \tag{2-4}$$

Dado que c es una constante, si conocemos el valor f podemos encontrar λ y viceversa. Como regla práctica, cuando λ se da en metros y f en MHz, $\lambda f \approx 300$. Por ejemplo, las ondas de 100 MHz tienen una longitud aproximada de 3 metros, las ondas de 1000 MHz tienen una longitud de 0.3 metros y las ondas de 0.1 metros tienen una frecuencia de 3 000 MHz.

En la figura 2-10 se muestra el espectro electromagnético. Las porciones de radio, microondas, infrarrojo y luz visible del espectro se pueden utilizar para transmitir información mediante la modulación de la amplitud, frecuencia o fase de las ondas. La luz ultravioleta, los rayos X y los rayos gamma serían todavía mejores, debido a sus frecuencias más altas, pero son difíciles de producir y de modular, no se propagan bien entre edificios y son peligrosos para los seres vivos. Las bandas que se listan en la parte inferior de la figura 2-10 son los nombres oficiales de la ITU (Unión Internacional de Telecomunicaciones) y se basan en las longitudes de onda, por lo que la banda LF va de 1 a 10 km (aproximadamente de 30 a 300 kHz). Los términos LF, MF y HF se refieren a las frecuencias baja, media y alta, respectivamente. Está claro que al asignar los nombres nadie esperaba rebasar los 10 MHz, por lo que las bandas más altas se denominaron después como bandas de muy, ultra, súper, extrema y tremendamente alta frecuencia. Más allá de eso ya no hay nombres, pero podrían sonar bien las designaciones de increíble, asombrosa y prodigiosamente alta frecuencia (IHF, AHF y PHF).

Sabemos de Shannon [ecuación (2-3)] que la cantidad de información que puede transportar una señal como una onda electromagnética depende de la potencia recibida y es proporcional a su ancho de banda. De la figura 2-10 debe quedar ahora claro por qué a las personas que trabajan en redes les gusta tanto la fibra óptica. Hay muchos GHz de ancho de banda disponibles que se pueden aprovechar para la transmisión de datos en la banda de microondas, e incluso más en la fibra debido a que está más a la derecha en nuestra escala logarítmica. Como ejemplo considere la banda de 1.30 micras de la figura 2-7, que tiene una anchura de 0.17 micras. Si utilizamos la ecuación (2-4) para buscar las frecuencias inicial y final a partir de las longitudes de onda inicial y final, obtenemos un rango de frecuencia aproximado de 30 000 GHz. Con una relación de señal-ruido razonable de 10 dB, esto sería 300 Tbps.

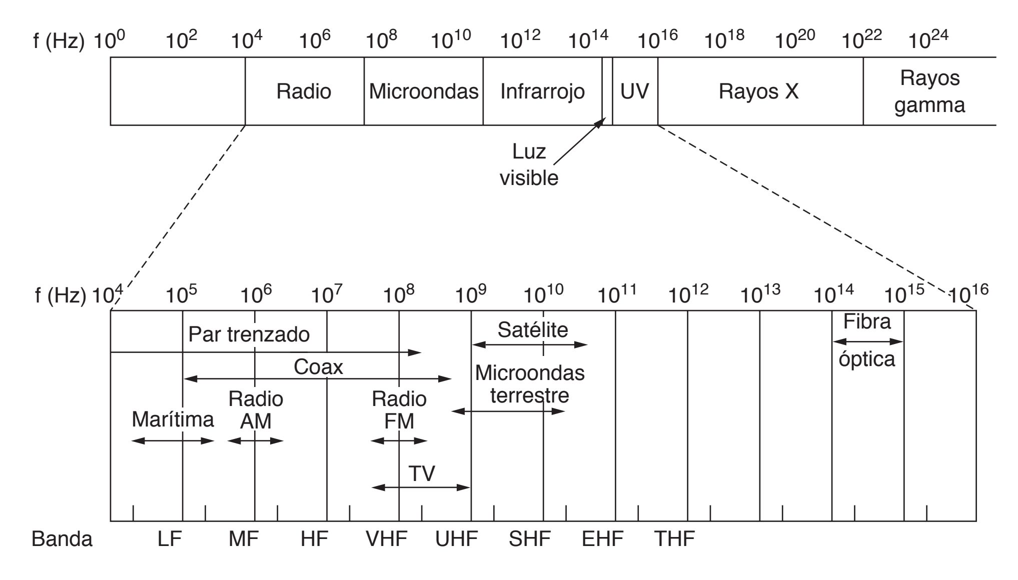


Figura 2-10. El espectro electromagnético y sus usos para comunicaciones.

La mayoría de las transmisiones utilizan una banda de frecuencia relativamente estrecha (por decir, $\Delta f / f \ll 1$). Concentran sus señales en esta banda estrecha para usar el espectro con eficiencia y obtienen tasas de datos razonables al transmitir con suficiente potencia. Pero en algunos casos se utiliza una banda ancha, con tres variaciones. En el **espectro disperso con salto de frecuencia**, el transmisor salta de frecuencia en frecuencia cientos de veces por segundo. Es popular en la comunicación militar, ya que hace a las transmisiones difíciles de detectar y casi imposibles de bloquear. También ofrece una buena resistencia al desvanecimiento multitrayectoria y a la interferencia de banda estrecha, ya que el receptor no se detendrá en una frecuencia dañada el tiempo suficiente como para detener la comunicación. Esta robustez la hace útil para las partes atestadas del espectro, como las bandas ISM que veremos más adelante. Bluetooth y las versiones anteriores de 802.11 son ejemplos del uso comercial de esta técnica.

Como una curiosa referencia, la técnica fue coinventada por la diosa del sexo de nacionalidad austriaca Hedy Lamarr, la primera mujer en aparecer desnuda en una película cinematográfica (el filme checoslovaco de 1933 de nombre *Extase*). Su primer esposo fue un fabricante de armamentos que le contó lo fácil que era bloquear las señales de radio que se utilizaban en ese entonces para controlar los torpedos. Cuando ella descubrió que su esposo estaba vendiendo armas a Hitler, quedó horrorizada, se disfrazó como sirvienta para escapar de él y voló a Hollywood para continuar su carrera como actriz de cine. En su tiempo libre inventó el salto de frecuencias para ayudar al esfuerzo bélico de los aliados. Su esquema utilizaba 88 frecuencias, el número de teclas (y frecuencias) de un piano. Por su invención, ella y su amigo George Antheil (compositor musical), recibieron la patente 2 292 387 de Estados Unidos. Sin embargo, no pudieron convencer a la marina de aquel país de que su invento tenía un uso práctico y nunca recibieron regalías. No fue sino años después de que expiró la patente cuando se hizo popular.

Hay una segunda forma de espectro disperso conocida como **espectro disperso de secuencia directa**, la cual utiliza una secuencia de códigos para dispersar los datos sobre una banda de frecuencia ancha. Su uso comercial es muy popular como una forma espectralmente eficiente de permitir que múltiples señales compartan la misma banda de frecuencia. Estas señales pueden recibir distintos códigos, un método conocido como **CDMA** (**Acceso Múltiple por División de Código**, del inglés *Code Division Multiple Access*) que veremos más adelante en este capítulo. En la figura 2-11 se muestra este método en contraste con el salto de frecuencias. CDMA forma la base de las redes de telefonía móvil 3G y también se utiliza en sistemas **GPS** (**Sistema de Posicionamiento Global**, del inglés *Global Positioning System*). Incluso con códigos distintos, el espectro disperso de secuencia directa (al igual que el espectro disperso de salto de frecuencia) puede tolerar la interferencia de banda estrecha y el desvanecimiento multitrayectoria, ya que sólo se pierde una fracción de la señal deseada. Se utiliza para desempeñar esta función en las redes LAN inalámbricas 802.11b anteriores. Si desea leer una historia fascinante y detallada sobre las comunicaciones de espectro disperso, consulte a Scholtz (1982).

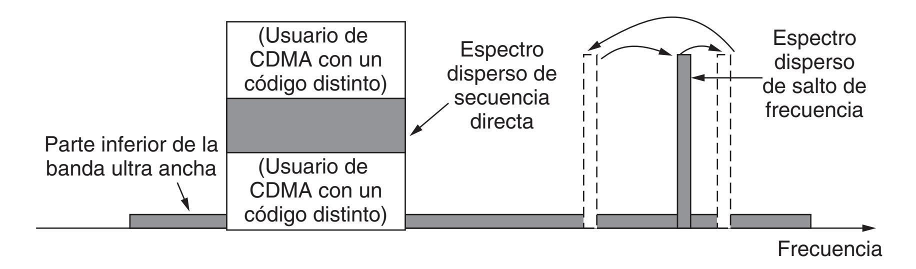


Figura 2-11. Comunicaciones de espectro disperso y Banda Ultra-Ancha (UWB).

Un tercer método de comunicación con una banda más ancha es la comunicación **UWB** (**Banda Ultra-Ancha**). La UWB envía una serie de pulsos rápidos, los cuales varían sus posiciones para comunicar

la información. Las transiciones rápidas conducen a una señal que se dispersa finamente sobre una banda de frecuencia muy amplia. UWB se define como señales que tienen un ancho de banda de por lo menos 500 MHz o de al menos 20% de la frecuencia central de su banda de frecuencia. En la figura 2-11 también se muestra la UWB. Con todo este ancho de banda, la UWB tiene el potencial para comunicarse a tasas altas. Como se dispersa a través de una banda amplia de frecuencias, puede tolerar una cantidad considerable de interferencia relativamente fuerte que provenga de otras señales de banda estrecha. Otra cuestión de igual importancia es que como la UWB tiene muy poca energía en cualquier frecuencia dada cuando se utiliza para la transmisión de corto rango, no provoca una interferencia dañina a todas esas otras señales de radio de banda estrecha. Se dice que **subyace debajo** de las otras señales. Gracias a esta pacífica coexistencia se utiliza en redes PAN inalámbricas que operan hasta a 1 Gbps, aunque el éxito comercial ha sido mezclado. También se puede usar para tomar imágenes a través de objetos sólidos (suelo, paredes y cuerpos) o como parte de los sistemas de ubicación precisos.

Ahora veremos cómo se utilizan las diversas partes del espectro electromagnético de la figura 2-11, empezando por la radio. Supongamos que todas las transmisiones utilizan una banda estrecha de frecuencia, a menos que se indique lo contrario.

2.3.2 Radiotransmisión

Las ondas de radio frecuencia (RF) son fáciles de generar, pueden recorrer distancias largas y penetrar edificios con facilidad, de modo que son muy utilizados en la comunicación, tanto en interiores como en exteriores. Las ondas de radio también son omnidireccionales, lo cual significa que viajan en todas direcciones desde la fuente, por lo que el transmisor y el receptor no tienen que estar alineados físicamente.

Algunas veces la radio omnidireccional es buena, pero otras no lo es tanto. En la década de 1970, General Motors decidió equipar a todos sus Cadillacs nuevos con frenos antibloqueo controlados por computadora. Cuando el conductor pisaba el pedal del freno, la computadora accionaba los frenos para activarlos y desactivarlos en vez de bloquearlos con firmeza. Un buen día, un patrullero de las carreteras de Ohio encendió su nuevo radio móvil para llamar a la estación de policía, cuando de repente el Cadillac que iba junto a él empezó a comportarse como un potro salvaje. Cuando el oficial detuvo el auto, el conductor alegó que no había hecho nada y que el auto se había vuelto loco.

Con el tiempo empezó a surgir un patrón: algunas veces los Cadillacs se volvían locos, pero sólo en las principales carreteras de Ohio y sólo cuando alguna patrulla de caminos estaba cerca. Durante mucho tiempo, General Motors no pudo comprender por qué los Cadillacs funcionaban bien en todos los demás estados e incluso en los caminos secundarios de Ohio. Después de que emprendieron una búsqueda extensa descubrieron que el cableado de los Cadillacs formaba una excelente antena para la frecuencia utilizada por el nuevo sistema de radio de las patrullas de caminos de Ohio.

Las propiedades de las ondas de radio dependen de la frecuencia. A bajas frecuencias, las ondas de radio cruzan bien los obstáculos, pero la potencia se reduce drásticamente a medida que se aleja de la fuente (por lo menos tan rápido como $1/r^2$ en el aire). A esta atenuación se le conoce como **pérdida de trayectoria**. A frecuencias altas, las ondas de radio tienden a viajar en línea recta y rebotan en los obstáculos. La pérdida de trayectoria reduce aún más la potencia, aunque la señal recibida también puede depender en gran parte de las reflexiones. Las ondas de radio de alta frecuencia también son absorbidas por la lluvia y otros obstáculos en mayor grado que las de baja frecuencia. En todas las frecuencias las ondas de radio están sujetas a interferencia de los motores y demás equipos eléctricos.

Es interesante comparar la atenuación de las ondas de radio con la de las señales en los medios guiados. Con la fibra, el cable coaxial y el par trenzado, la señal se reduce en la misma fracción por distancia de unidad, por ejemplo 20 dB por cada 100 m para el par trenzado. Con la radio, la señal se reduce en la misma fracción a medida que se duplica la distancia, por ejemplo 6 dB por cada vez que se duplique

la distancia en el espacio libre. Este comportamiento indica que las ondas de radio pueden recorrer grandes distancias y, en consecuencia, la interferencia entre usuarios es un problema. Por esta razón, es común que los gobiernos regulen estrictamente el uso de los radiotransmisores, con algunas excepciones notables que veremos más adelante en este capítulo.

En las bandas VLF, LF y MF las ondas de radio siguen la curvatura de la Tierra, como se muestra en la figura 2-12(a). Estas ondas se pueden detectar quizás a 1000 km en las frecuencias más bajas, y a menos distancia en las frecuencias más altas. La difusión de radio AM utiliza la banda MF; por esta razón las ondas terrestres de las estaciones de radio AM de Boston no se pueden escuchar con facilidad en Nueva York. Las ondas de radio en estas bandas pasan por los edificios fácilmente, razón por la cual los radios portátiles funcionan en interiores. El principal problema al usar estas bandas para la comunicación de datos es su bajo ancho de banda [vea la ecuación (2-4)].

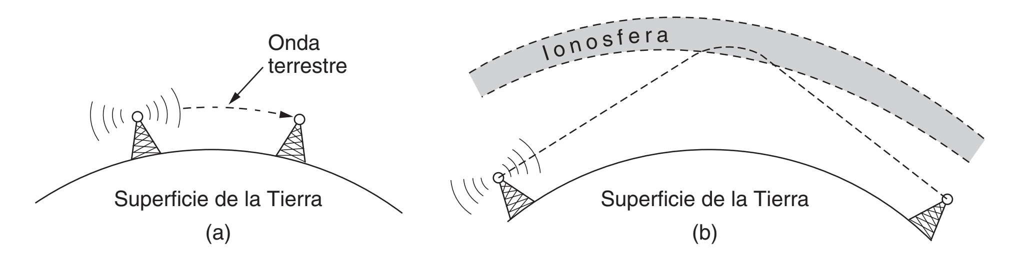


Figura 2-12. (a) En las bandas VLF, LF y MF, las ondas de radio siguen la curvatura de la Tierra. (b) En la banda HF, rebotan en la ionosfera.

En las bandas HF y VHF, las ondas terrestres tienden a ser absorbidas por la Tierra. Sin embargo, las ondas que llegan a la ionosfera (una capa de partículas cargadas que rodean la Tierra a una altura de 100 a 500 km) se refractan y se envían de vuelta a nuestro planeta, como se muestra en la figura 2-12(b). Bajo ciertas condiciones atmosféricas, las señales pueden rebotar varias veces. Los operadores de las bandas de radio aficionados utilizan estas bandas para conversar a larga distancia. El ejército también se comunica en las bandas HF y VHF.

2.3.3 Transmisión por microondas

Por encima de los 100 MHz, las ondas viajan en línea recta y en consecuencia, se pueden enfocar en un haz estrecho. Al concentrar toda la energía en un pequeño haz por medio de una antena parabólica (como el tan conocido plato de TV por satélite) se obtiene una relación señal-ruido mucho más alta, pero las antenas transmisora y receptora deben estar alineadas entre sí con precisión. Además, esta direccionalidad permite que varios transmisores alineados en fila se comuniquen con varios receptores sin interferencia, siempre y cuando se sigan ciertas reglas de espacio mínimo. Antes de la fibra óptica, estas microondas formaron durante décadas el corazón del sistema de transmisión telefónica de larga distancia. De hecho, la empresa MCI (uno de los primeros competidores de AT&T después de su liberación) construyó todo su sistema a partir de comunicaciones por microondas que iban de torre en torre ubicadas a decenas de kilómetros una de la otra. Incluso el nombre de la empresa reflejaba esta cuestión (MCI representa a Microwave Communications, Inc.). Después MCI cambió a la fibra óptica y por medio de una extensa serie de fusiones corporativas y bancarrotas en la reestructuración de las telecomunicaciones, se volvió parte de Verizon.

Puesto que las microondas viajan en línea recta, si las torres están demasiado separadas, la Tierra se interpondrá en el camino (imagine un enlace de Seattle a Ámsterdam). Por ende se necesitan repetidores

periódicos. Entre más altas sean las torres, más separadas pueden estar. La distancia entre repetidores se eleva en forma muy aproximada a la raíz cuadrada de la altura de la torre. Si tenemos torres de 100 metros de altura, los repetidores pueden estar separados a 80 km de distancia.

A diferencia de las ondas de radio a frecuencias más bajas, las microondas no pueden atravesar bien los edificios. Además, aun cuando el haz puede estar bien enfocado en el transmisor, hay cierta divergencia en el espacio. Algunas ondas pueden refractarse en las capas atmosféricas más bajas y tardar un poco más en llegar que las ondas directas. Estas ondas retrasadas pueden llegar desfasadas con la onda directa y cancelar así la señal. A este efecto se le llama **desvanecimiento por multitrayectorias** y representa a menudo un problema grave que depende del clima y de la frecuencia. Algunos operadores mantienen el 10% de sus canales inactivos como repuestos para activarlos cuando el desvanecimiento por multitrayectorias cancela en forma temporal alguna banda de frecuencia.

La creciente demanda de espectro obliga a los operadores a usar frecuencias aún más altas. Ahora las bandas de hasta 10 GHz son de uso rutinario, pero con las de casi 4 GHz se presenta un nuevo problema: la absorción por el agua. Estas ondas tienen sólo unos centímetros de longitud y la lluvia las absorbe. Este efecto sería benéfico si quisiéramos construir un enorme horno de microondas para rostizar las aves de paso, pero para la comunicación representa un problema grave. Al igual que con el desvanecimiento por multitrayectorias, la única solución es interrumpir los enlaces afectados por la lluvia y encaminar las señales a su alrededor.

En resumen, la comunicación por microondas se utiliza tanto para la comunicación telefónica de larga distancia, los teléfonos móviles, la distribución de la televisión y otros usos, que ha provocado una escasez de espectro. Esta tecnología tiene varias ventajas clave respecto a la fibra. La principal es que no se necesita derecho de paso para tender los cables. Con sólo comprar un pequeño terreno cada 50 km y construir en él una torre de microondas, se puede pasar por alto el sistema telefónico en su totalidad. Ésta es la forma en que MCI logró establecerse como una nueva compañía telefónica de larga distancia con tanta rapidez (Sprint, otro de los primeros competidores de AT&T después de su liberación, siguió un camino distinto: fue fundada por la empresa ferrocarrilera Southern Pacific Railroad, que ya poseía una gran cantidad de derechos de paso y sólo tuvo que enterrar la fibra junto a las vías).

Las microondas son también relativamente económicas. Puede ser más barato erigir dos torres sencillas (que pueden ser tan sólo unos postes grandes con cuatro cables de retén) y poner antenas en cada una de ellas, que enterrar 50 km de fibra a través de un área urbana congestionada o sobre una montaña, y también puede ser más económico que rentar la fibra de la compañía telefónica, en especial si ésta no ha recuperado por completo la inversión por el cobre que quitó al instalar la fibra.

Las políticas del espectro electromagnético

Para evitar el caos total, existen acuerdos nacionales e internacionales en cuanto a quién puede usar ciertas frecuencias. Como todos quieren una tasa más alta de transferencia de datos, todos quieren más espectro. Los gobiernos nacionales asignan el espectro para la radio AM y FM, la televisión y los teléfonos móviles, así como para las compañías telefónicas, la policía, las comunicaciones marítimas, la navegación, el ejército, el gobierno y muchos otros usuarios competidores. A nivel mundial, una agencia de la ITU-R (WRC) trata de coordinar esta asignación de modo que se puedan fabricar dispositivos que funcionen en varios países. Sin embargo, los países no están obligados a seguir las recomendaciones de la ITU-R, por lo que la FCC (Comisión Federal de Comunicaciones), que se encarga de la asignación para Estados Unidos, en ocasiones ha rechazado las recomendaciones de la ITU-R (por lo general debido a que tenían que obligar a algún grupo con poder político a ceder una parte del espectro).

Incluso cuando se haya asignado una parte del espectro para cierto uso, como los teléfonos móviles, se debe determinar qué empresa portadora puede utilizar qué frecuencias. En el pasado se utilizaban tres algoritmos. El más viejo se conoce como **concurso de méritos** *(beauty contest)*; en este algoritmo cada

empresa portadora tiene que explicar por qué su propósito es más útil para el interés público. Después, los funcionarios de gobierno deciden cuál de todas esas historias los convence más. El hecho de que un funcionario de gobierno pueda otorgar una propiedad con valor de miles de millones de dólares a su compañía favorita conduce con frecuencia al soborno, la corrupción, el nepotismo y cosas peores. Lo que es más, incluso un funcionario de gobierno escrupulosamente honesto que pensara que una empresa extranjera podría realizar un mejor trabajo que cualquiera de las compañías nacionales, tendría mucho qué explicar.

Esta observación condujo al algoritmo 2: llevar a cabo un **sorteo** entre las compañías interesadas. El problema con esta idea es que pueden entrar al sorteo empresas que no tengan interés en utilizar el espectro. Por decir, si un restaurante de comida rápida o una cadena de tiendas de zapatos ganan, puede revender el espectro a una portadora para obtener una enorme ganancia sin ningún riesgo.

Este proceso ha sido criticado con severidad por muchos, lo cual condujo al algoritmo 3: **subastar** el ancho de banda al mejor postor. Cuando el gobierno británico subastó las frecuencias necesarias para los sistemas móviles de tercera generación en el año 2000, esperaba obtener cerca de $4 mil millones. En realidad recibió cerca de $40 mil millones debido a que las empresas portadoras cayeron en la desesperación, muertas de miedo de dejar pasar la oportunidad. Este suceso despertó la avaricia de los gobiernos vecinos y los inspiró a realizar sus propias subastas. Esto funcionó, pero a la vez algunas de las empresas portadoras quedaron con deudas enormes que las llevaron cerca de la bancarrota. Aun en los mejores casos, se requerirán muchos años para recuperar la inversión en la licencia.

Una metodología completamente distinta a la asignación de frecuencias es la de no asignarlas en lo absoluto. En vez de ello hay que dejar que todos transmitan a voluntad, pero que regulen la potencia utilizada de modo que las estaciones tengan un rango tan corto que no interfieran entre sí. En consecuencia, la mayoría de los gobiernos han separado ciertas bandas de frecuencia, llamadas bandas **ISM** (**Industriales, Científicas, Médicas**, del inglés *Industrial, Scientific, Medical*), para un uso sin necesidad de licencia. Los dispositivos para abrir puertas de cocheras, los teléfonos inalámbricos, los juguetes de radiocontrol, los ratones inalámbricos y muchos otros electrodomésticos inalámbricos utilizan las bandas ISM. Para minimizar la interferencia entre estos dispositivos no coordinados, la FCC exige que todos los dispositivos en las bandas ISM limiten su potencia de transmisión (por ejemplo, a 1 watt) y utilicen técnicas para dispersar sus señales a través de un rango de frecuencias. Tal vez los dispositivos también tengan que tomar las precauciones necesarias para evitar interferencias con las instalaciones de radar.

La ubicación de estas bandas varía de un país a otro. Por ejemplo, en Estados Unidos las bandas que utilizan los dispositivos de red en la práctica sin requerir una licencia de FCC se muestran en la figura 2-13. La banda de 900 MHz se utilizaba para las primeras versiones de 802.11, pero está muy llena. La banda de 2.4 GHz está disponible en la mayoría de los países y se utiliza mucho para 802.11b/g y Bluetooth, aunque está sujeta a las interferencias de los hornos de microondas y las instalaciones de radar. La parte del espectro correspondiente a los 5 GHz incluye bandas **U-NII** (**Infraestructura de Información Nacional sin Licencia**, del inglés *Unlicensed National Information Infrastructure*). Las bandas de 5 GHz tienen muy poco desarrollo pero, como tienen el mayor ancho de banda y son utilizadas por el estándar 802.11a, están ganando popularidad rápidamente.

Las bandas sin licencia tuvieron un tremendo éxito durante la última década. La habilidad de utilizar el espectro con libertad ha desencadenado una gran innovación en las redes LAN y PAN inalámbricas, tal y como se evidencia a través del desarrollo extendido de tecnologías como 802.11 y Bluetooth. Para continuar con esta innovación se necesita más espectro. Un excitante acontecimiento en Estados Unidos es la decisión de la FCC en el 2009 de permitir el uso sin necesidad de licencia de los **espacios en blanco** alrededor de los 700 MHz. Los espacios en blanco son bandas de frecuencia que están asignadas pero no se utilizan en forma local. La transición de la difusión de televisión analógica completamente digital en

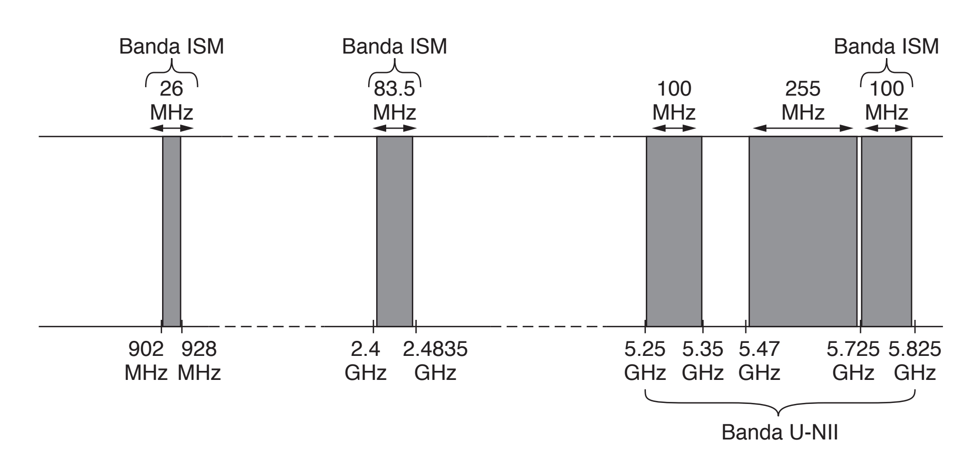


Figura 2-13. Las bandas ISM y U-NII que los dispositivos inalámbricos utilizan en Estados Unidos.

Estados Unidos en el 2010 liberó los espacios en blanco alrededor de los 700 MHz. La única dificultad es que, para usar los espacios en blanco, los dispositivos sin licencia deben detectar cualquier transmisor cercano con licencia, incluyendo los micrófonos inalámbricos, que tengan el derecho de usar primero la banda de frecuencia.

Hay otra oleada de actividad relacionada con la banda de 60 GHz. La FCC abrió la banda de 57 a 64 GHz para la operación sin licencia en 2001. Este rango es una enorme porción del espectro, mayor a todas las demás bandas ISM combinadas, por lo que puede soportar el tipo de redes de alta velocidad que se necesitarían para transmitir TV de alta definición en flujo continuo por el aire hasta los hogares. A los 60 GHz, el oxígeno absorbe las ondas de radio. Esto significa que las señales no se propagan largas distancias, por lo que son ideales para las redes de corto alcance. Al principio, las altas frecuencias (los 60 GHz se encuentran en la banda de muy alta frecuencia o "milimétrica", justo debajo de la radiación infrarroja) representaban un reto para los fabricantes de equipos, pero ahora ya hay productos en el mercado.

2.3.4 Transmisión infrarroja

Las ondas infrarrojas no guiadas se usan mucho para la comunicación de corto alcance. El control remoto de los televisores, grabadoras de video y estéreos utilizan comunicación infrarroja. Son relativamente direccionales, económicos y fáciles de construir, pero tienen un gran inconveniente: no atraviesan objetos sólidos (pruebe pararse entre el control remoto y su televisión, y vea si aún funciona). En general, conforme pasamos de la radio de onda larga hacia la luz visible, las ondas se comportan cada vez más como la luz y cada vez menos como la radio.

Por otro lado, el hecho de que las ondas infrarrojas no atraviesen bien las paredes sólidas también es una ventaja. Esto significa que un sistema infrarrojo en un cuarto de un edificio no interferirá con un sistema similar en cuartos o edificios adyacentes; no podrá controlar la televisión de su vecino con su control remoto. Además, la seguridad de los sistemas infrarrojos contra el espionaje es mejor que la de los sistemas de radio, precisamente por esta razón. Por ende, no se necesita licencia gubernamental para operar un sistema infrarrojo, en contraste con los sistemas de radio, que deben contar con licencia excepto las bandas ISM. La comunicación infrarroja tiene un uso limitado en el escritorio; por ejemplo, para conectar computadoras portátiles e impresoras mediante el estándar **IrDA** (**Asociación de Datos por Infrarrojo**, del inglés *Infrared Data Association*), aunque no es un protagonista importante en el juego de las comunicaciones.

2.3.5 Transmisión por ondas de luz

La señalización óptica sin guías, también conocida como **óptica de espacio libre**, se ha utilizado durante siglos. Paul Revere utilizó señalización óptica binaria desde la vieja Iglesia del Norte justo antes de su famoso viaje. Una aplicación más moderna es conectar las redes LAN de dos edificios mediante láser montados en sus azoteas. La señalización óptica mediante láser es de naturaleza unidireccional, por lo que cada extremo necesita su propio láser y su propio fotodetector. Este esquema ofrece un ancho de banda muy alto a un costo muy bajo, además de ser relativamente seguro debido a que es difícil intervenir un haz tan estrecho. También es relativamente fácil de instalar y, a diferencia de las microondas, no requiere una licencia de la FCC.

La ventaja del láser, un haz muy estrecho, es también su debilidad en este caso. Para apuntar un rayo láser de 1 mm de anchura a un blanco del tamaño de la punta de un alfiler a 500 metros de distancia, se requiere la puntería de una Annie Oakley moderna. Por lo general se añaden lentes al sistema para desenfocar ligeramente el rayo. Para dificultar aún más las cosas, los cambios en el viento y la temperatura pueden distorsionar el rayo, además de que los rayos láser no pueden penetrar la lluvia o la niebla densa, aunque por lo general funcionan bien en días soleados. Sin embargo, muchos de estos factores no representan un problema a la hora de conectar dos naves espaciales.

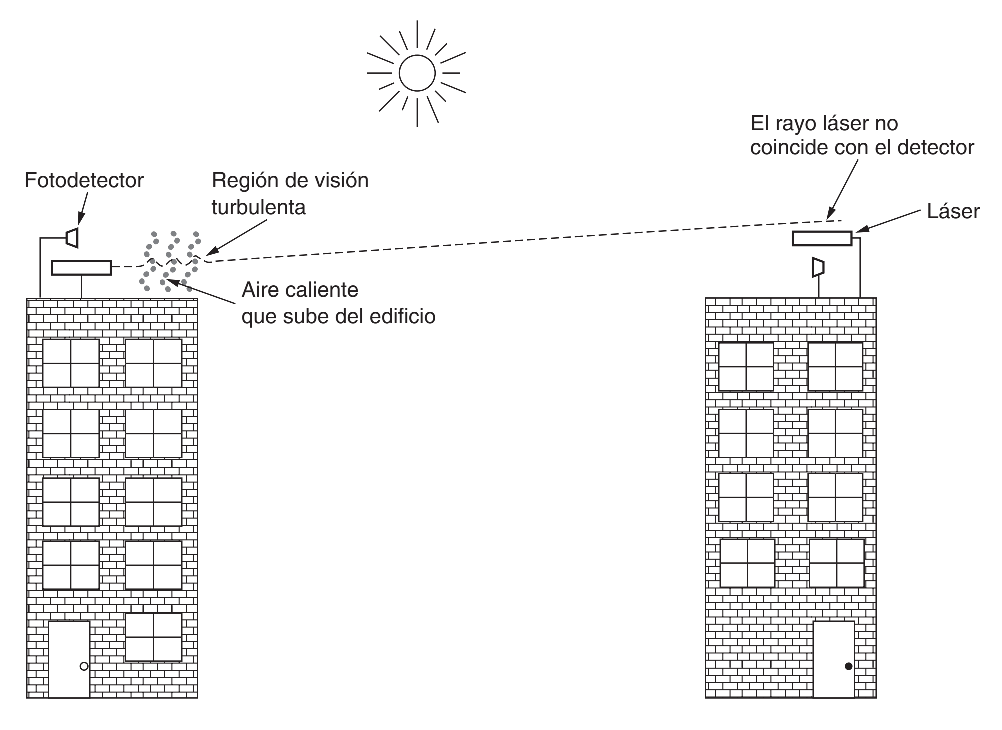


Figura 2-14. Las corrientes de convección pueden interferir con los sistemas de comunicación por láser. Aquí se ilustra un sistema bidireccional con dos láser.

Uno de los autores (AST) asistió en una ocasión a una conferencia en un moderno hotel de Europa, en donde los organizadores de la conferencia tuvieron la atención de proveer un cuarto lleno de terminales para que los asistentes pudieran leer su correo electrónico durante las presentaciones aburridas. Puesto que la PTT local no estaba dispuesta a instalar un gran número de líneas telefónicas sólo para tres días,

los organizadores colocaron un láser en el techo y lo apuntaron al edificio de ciencias computacionales de su universidad, que se encontraba a unos cuantos kilómetros de distancia. Lo probaron la noche anterior a la conferencia y funcionó a la perfección. A las 9:00 a.m. de un día brillante y soleado, el enlace falló por completo y permaneció caído todo el día. El patrón se repitió durante dos días en forma idéntica. No fue sino hasta después de la conferencia que los organizadores descubrieron el problema: el calor del Sol durante el día provocaba corrientes de convección que se elevaban desde el techo del edificio, como se muestra en la figura 2-14. Este aire turbulento desviaba el rayo y lo hacía bailar alrededor del detector, en forma muy parecida a un camino resplandeciente en un día caluroso. La moraleja en este caso es que, para trabajar bien tanto en condiciones difíciles como ideales, hay que diseñar los enlaces ópticos sin guía con el suficiente margen de error.

La comunicación óptica sin guía puede parecer una tecnología de redes exótica en la actualidad, pero pronto puede llegar a ser más frecuente. Estamos rodeados por cámaras (que detectan la luz) y pantallas (que emiten luz mediante el uso de LED y otras tecnologías). La comunicación de datos se puede disponer en capas encima de estas pantallas si se codifica la información en el patrón que hace que los LED se enciendan y apaguen, y que está por debajo del umbral de la percepción humana. Esta forma de comunicarse con luz visible es segura por naturaleza, además de que crea una red de baja velocidad en los alrededores inmediatos de la pantalla. Esto podría permitir todo tipo de escenarios computacionales elegantes y ubicuos. Las luces destellantes en los vehículos de emergencia podrían alertar a los semáforos cercanos y a los vehículos para ayudar a dejar libre el camino. Los anuncios informativos podrían difundir mapas. Incluso las luces de las festividades podrían difundir canciones sincronizadas con su pantalla.

2.4 SATÉLITES DE COMUNICACIÓN

En la década de 1950 y a principios de la década de 1960, las personas trataban de establecer sistemas de comunicación mediante el rebote de señales sobre globos meteorológicos. Por desgracia, las señales que se recibían eran demasiado débiles como para darles un uso práctico. Después, la marina de Estados Unidos observó un tipo de globo meteorológico permanente en el cielo (la Luna), de modo que construyó un sistema operacional para la comunicación de barcos con la costa mediante señales que rebotaban de la Luna.

El avance en el campo de la comunicación celestial tuvo que esperar hasta que se lanzó el primer satélite de comunicaciones. La diferencia clave entre un satélite artificial y uno real es que el primero puede amplificar las señales antes de enviarlas de regreso, convirtiendo una extraña curiosidad en un poderoso sistema de comunicaciones.

Los satélites de comunicaciones tienen ciertas propiedades interesantes que los hacen atractivos para muchas aplicaciones. En su forma más simple, podemos considerar un satélite de comunicaciones como un enorme repetidor de microondas en el cielo que contiene varios **transpondedores**, cada uno de los cuales escucha en cierta porción del espectro, amplifica la señal entrante y después la retransmite en otra frecuencia para evitar interferencia con la señal entrante. Este modo de operación se llama **tubo doblado**. Se puede agregar un procesamiento digital para manipular o redirigir por separado los flujos de datos en toda la banda, o el satélite puede recibir información digital y retransmitirla. Esta forma de regeneración de señales mejora el desempeño si se le compara con un tubo doblado, ya que el satélite no amplifica el ruido en la señal que va hacia arriba. Los haces que descienden pueden ser amplios y cubrir una fracción considerable de la superficie de la Tierra, o pueden ser estrechos y cubrir un área de unos cuantos cientos de kilómetros de diámetro.

De acuerdo con la ley de Kepler, el periodo orbital de un satélite varía según el radio de la órbita a la 3/2 potencia. Entre más alto esté el satélite, mayor será el periodo. Cerca de la superficie de la Tierra, el periodo es de aproximadamente 90 minutos. En consecuencia, los satélites con órbitas bajas salen del rango de visión muy rápido, de modo que muchos de ellos deben proveer una cobertura continua y las

antenas terrestres deben rastrearlos. A una altitud aproximada de 35 800 km, el periodo es de 24 horas. A una altitud de 384 000 km el periodo es de cerca de un mes, como puede atestiguar cualquiera que haya observado la Luna con regularidad.

El periodo de un satélite es importante, pero no es la única razón para determinar en dónde colocarlo. Otra cuestión es la presencia de los cinturones de Van Allen: capas de partículas altamente cargadas, atrapadas por el campo magnético de la Tierra. Cualquier satélite que volara dentro de los cinturones quedaría destruido casi al instante debido a las partículas. Estos factores condujeron a tres regiones en las que se pueden colocar los satélites de forma segura. En la figura 2-15 se muestran estas regiones con algunas de sus propiedades. A continuación describiremos brevemente los satélites que habitan en cada una de estas regiones.

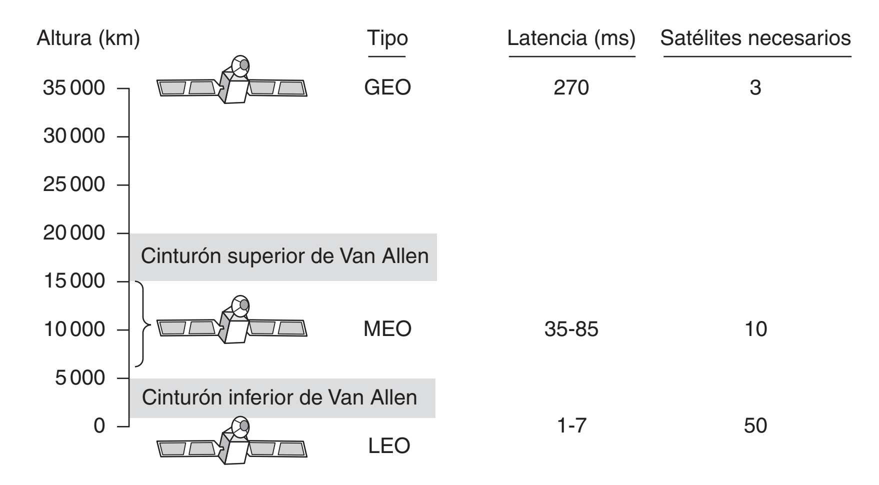


Figura 2-15. Satélites de comunicaciones y algunas de sus propiedades, incluyendo la altitud sobre la Tierra, el tiempo de retardo de viaje redondo y la cantidad de satélites necesarios para una cobertura global.

2.4.1 Satélites geoestacionarios

En 1945, el escritor de ciencia ficción Arthur C. Clarke calculó que un satélite con una altitud de 35 800 km en una órbita ecuatorial circular parecería estar inmóvil en el cielo, por lo que no habría la necesidad de rastrearlo (Clarke, 1945). Pasó a describir un sistema completo de comunicaciones que utilizaba estos **satélites geoestacionarios** (tripulados), incluyendo las órbitas, los paneles solares, las frecuencias de radio y los procedimientos de lanzamiento. Por desgracia concluyó que los satélites no eran prácticos debido a la imposibilidad de poner en órbita amplificadores de tubos de vacío frágiles y que consumían una gran cantidad de energía, por lo que nunca profundizó sobre esta idea, aunque escribió algunas historias de ciencia ficción sobre el tema.

La invención del transistor cambió todo eso; el primer satélite de comunicación artificial llamado Telstar se lanzó en julio de 1962. Desde entonces, los satélites de comunicación se convirtieron en un negocio multimillonario y el único aspecto del espacio exterior que se ha vuelto muy rentable. Estos satélites que vuelan a grandes alturas se conocen comúnmente como satélites **GEO** (**Órbita Terrestre Geoestacionaria**, del inglés *Geostationary Earth Orbit*).

Con la tecnología actual, es poco sensato tener satélites geoestacionarios separados a menos de 2 grados en el plano ecuatorial de 360 grados para evitar interferencias. Con una separación de 2 grados

sólo puede haber 360/2 = 180 de estos satélites en el cielo a la vez. Sin embargo, cada transpondedor puede usar múltiples frecuencias y polarizaciones para incrementar el ancho de banda disponible.

Para evitar un caos total en el cielo, la ITU se encarga de la asignación de espacio orbital. Este proceso es altamente político, con casos en donde países que apenas acaban de salir de la edad de piedra exigen "sus" espacios orbitales (con el propósito de rentarlos al mejor postor). Sin embargo, otros países afirman que los derechos de propiedad nacional no se extienden hasta la Luna y que ningún país tiene el derecho legal sobre los espacios orbitales encima de su territorio. Para hacer más grande la pelea, la telecomunicación comercial no es la única aplicación. Las difusoras de televisión, los gobiernos y el ejército también desean una parte del espacio orbital.

Los satélites modernos pueden ser muy grandes y pesar más de 5 000 kg, además de que consumen varios kilowatts de energía eléctrica producida por los paneles solares. Los efectos de la gravedad solar, lunar y planetaria tienden a alejarlos de sus espacios orbitales y orientaciones asignadas, un efecto que se contrarresta mediante motores de cohete integrados. Esta actividad de ajuste se conoce como **control de la posición orbital** (*station keeping*). Sin embargo, cuando se agota el combustible de los motores (por lo general después de casi 10 años), el satélite queda a la deriva y cae sin que se pueda hacer nada, de modo que debe ser desactivado. En un momento dado la órbita se deteriora, el satélite vuelve a entrar en la atmósfera y se quema (o en muy raras ocasiones, se estrella en la Tierra).

Los espacios orbitales no son el único motivo de discordia. Las frecuencias también son otro problema debido a que las transmisiones de los enlaces descendentes interfieren con los usuarios existentes de microondas. En consecuencia, la ITU ha asignado bandas de frecuencia específicas a los usuarios de satélites. Las principales se muestran en la figura 2-16. La banda C fue la primera en ser designada para el tráfico comercial por satélites. Hay dos rangos de frecuencia asignados a esta banda, el inferior para el tráfico de enlace descendente (proveniente del satélite) y el superior para el tráfico del enlace ascendente (que va al satélite). Para permitir que el tráfico viaje en ambos sentidos al mismo tiempo se requieren dos canales. Estos canales ya están sobresaturados debido a que también son utilizados por las portadoras comunes para los enlaces terrestres de microondas. Las bandas L y S se agregaron con base en un acuerdo internacional en el año 2000. Sin embargo, son estrechas y también están saturadas.

Banda	Enlace descendente	Enlace ascendente	Ancho de banda	Problemas
L	1.5 GHz	1.6 GHz	15 MHz	Bajo ancho de banda; saturada.
S	1.9 GHz	2.2 GHz	70 MHz	Bajo ancho de banda; saturada.
C	4.0 GHz	6.0 GHz	500 MHz	Interferencia terrestre.
Ku	11 GHz	14 GHz	500 MHz	Lluvia.
Ka	20 GHz	30 GHz	3500 MHz	Lluvia, costo del equipo.

Figura 2-16. Las principales bandas de satélites.

La siguiente banda más ancha disponible para las portadoras de telecomunicaciones comerciales es la banda **Ku** (**K inferior**, del inglés *K under*). Esta banda (aún) no está saturada; a sus frecuencias más altas los satélites pueden tener una separación mínima de 1 grado. Sin embargo, existe otro problema: la lluvia. El agua absorbe bien estas microondas cortas. Por fortuna, las tormentas fuertes por lo general son localizables, de modo que para resolver el problema se pueden usar varias estaciones terrestres separadas a grandes distancias en vez de usar sólo una, pero a costa de requerir antenas, cables y componentes electrónicos adicionales para permitir una conmutación rápida entre las estaciones. También se asignó ancho de banda en la banda **Ka** (**K superior**, del inglés *K above*) para el tráfico comercial de satélites, pero el

equipo necesario para utilizarla es costoso. Además de estas bandas comerciales, también existen muchas bandas gubernamentales y militares.

Un satélite moderno tiene cerca de 40 transpondedores, cada uno con un ancho de banda de 36 MHz. Por lo general cada transpondedor opera como un tubo doblado, pero los satélites recientes cuentan con capacidad de procesamiento integrada, lo cual les permite una operación más sofisticada. En los primeros satélites, la división de los transpondedores en canales era estática: el ancho de banda simplemente se dividía en bandas fijas de frecuencia. Hoy en día el haz de cada transpondedor se divide en ranuras de tiempo, en donde varios usuarios toman turnos. Más adelante estudiaremos estas dos técnicas (multiplexión por división de frecuencia y multiplexión por división de tiempo).

Los primeros satélites geoestacionarios tenían un solo haz espacial que iluminaba aproximadamente 1/3 de la superficie de la Tierra, a lo cual se le conoce como **huella**. Con la enorme reducción en el precio, tamaño y requerimientos de energía de los componentes microelectrónicos, se ha hecho posible una estrategia de difusión mucho más sofisticada. Cada satélite está equipado con múltiples antenas y múltiples transpondedores. Cada haz descendente se puede enfocar en una pequeña área geográfica, de manera que se pueden llevar a cabo varias transmisiones ascendentes y descendentes simultáneamente. Por lo general, estos denominados **haces puntuales** tienen una forma elíptica y pueden ser tan pequeños como de algunos cientos de kilómetros de diámetro. Por lo general, un satélite de comunicación para Estados Unidos tiene un haz amplio para los 48 estados contiguos, además de haces puntuales para Alaska y Hawai.

Un reciente acontecimiento en el mundo de los satélites de comunicaciones es el desarrollo de microestaciones de bajo costo, conocidas también como **VSAT** (**Terminales de apertura muy pequeña**, del inglés *Very Small Apertura Terminals*) (Abramson, 2000). Estas pequeñas terminales tienen antenas de 1 metro o menos (en comparación con las antenas GEO estándar de 10 m) y pueden emitir cerca de 1 watt de potencia. El enlace ascendente es generalmente bueno para soportar hasta 1 Mbps, pero el enlace descendente puede soportar por lo general hasta varios megabits/seg. La televisión vía satélite de difusión directa utiliza esta tecnología para la transmisión unidireccional.

En muchos sistemas VSAT, las microestaciones no tienen suficiente potencia como para comunicarse de manera directa unas con otras (a través del satélite, claro está). En vez de ello se necesita una estación especial terrestre (**hub** o **estación central**) con una antena grande y con mucha potencia para transmitir el tráfico entre las VSAT, como se muestra en la figura 2-17. En este modo de operación, el emisor o el receptor tienen una antena grande y un amplificador poderoso. La desventaja es un retardo más grande a cambio de tener estaciones más económicas para el usuario final.

Los VSAT tienen gran potencial en áreas rurales. No son muy apreciados, pero más de la mitad de la población mundial vive a más de una hora de distancia a pie del teléfono más cercano. Instalar cables telefónicos en miles de pequeñas aldeas es algo que está más allá de los presupuestos de la mayoría de los gobiernos del tercer mundo, pero instalar platos VSAT de 1 metro operados por celdas solares es algo muy factible. Los VSAT proveen la tecnología para cablear al mundo.

Los satélites de comunicación tienen varias propiedades que son radicalmente distintas a las de los enlaces terrestres de punto a punto. Para empezar, aun cuando las señales hacia y desde un satélite viajan a la velocidad de la luz (cerca de 300 000 km/seg), la larga distancia de viaje redondo introduce un retardo considerable para los satélites GEO. Dependiendo de la distancia entre el usuario y la estación terrestre, y de la elevación del satélite sobre el horizonte, el tiempo de tránsito de un extremo a otro está entre 250 y 300 mseg. Un valor común es 270 mseg (540 mseg para un VSAT con una estación central, o hub).

Para fines de comparación, los enlaces terrestres de microondas tienen un retardo de propagación de aproximadamente 3 μseg/km, y los enlaces de cable coaxial o fibra óptica tienen un retardo de casi 5 μseg/km. Los últimos son más lentos que los primeros debido a que las señales electromagnéticas viajan con más rapidez en el aire que en los materiales sólidos.

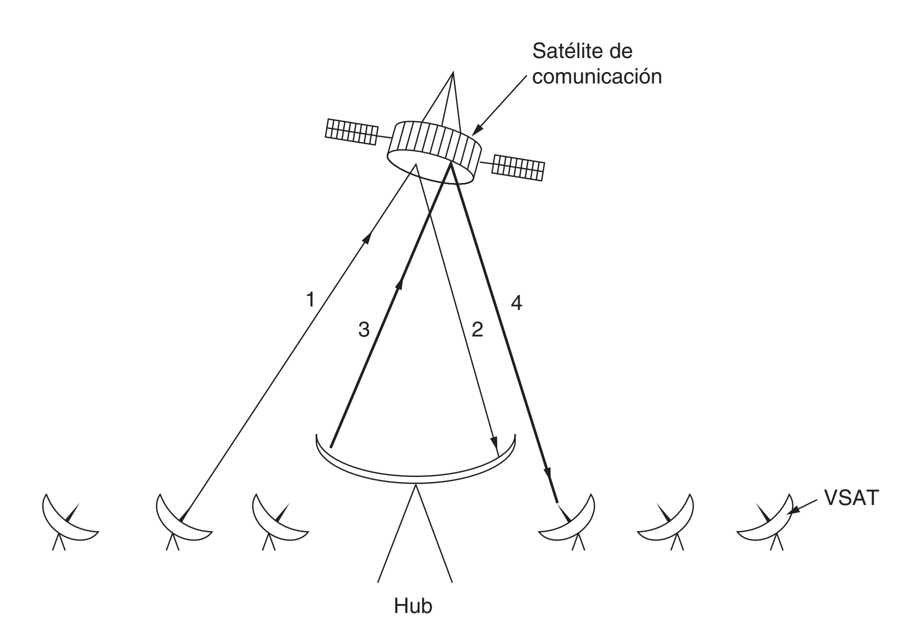


Figura 2-17. Terminales VSAT que utilizan una estación central o hub.

Otra propiedad importante de los satélites es que son medios de difusión por naturaleza. Cuesta lo mismo enviar un mensaje a miles de estaciones dentro de la huella de un transpondedor que enviarlo a una sola. Para algunas aplicaciones esta propiedad es muy útil. Por ejemplo, podríamos imaginar un satélite difundiendo páginas web populares a las cachés de una gran cantidad de computadoras dispersas sobre un área amplia. Aun cuando podemos simular la difusión mediante líneas de punto a punto, es probable que la difusión vía satélite sea más económica. Por otro lado, desde el punto de vista de la privacidad, los satélites son un completo desastre: todos pueden escucharlo todo. Es esencial el cifrado cuando se requiere seguridad.

Los satélites también tienen la propiedad de que el costo de transmitir un mensaje es independiente de la distancia a recorrer. Es lo mismo dar servicio a una llamada de un extremo a otro del océano que una llamada de un extremo a otro de la calle. Los satélites también tienen excelentes tasas de error y pueden implementarse casi al instante, una buena ventaja para las comunicaciones militares y de respuesta a los desastres.

2.4.2 Satélites de Órbita Terrestre Media (MEO)

En altitudes mucho más bajas entre los dos cinturones de Van Allen se encuentran los satélites **MEO** (**Orbita Terrestre Media**, del inglés *Medium-Earth Orbit*). Vistos desde la Tierra, se desvían lentamente en longitud y tardan cerca de seis horas en dar vuelta a la Tierra. Por ende, hay que rastrearlos a medida que se mueven por el cielo. Como tienen menor altura que los satélites GEO, producen una huella más pequeña en la Tierra y requieren transmisores menos poderosos para comunicarse. En la actualidad se utilizan para sistemas de navegación en vez de las telecomunicaciones, por lo que no daremos más detalles sobre ellos. La constelación de alrededor de 30 satélites **GPS** (**Sistema de Posicionamiento Global**, del inglés *Global Positioning System*) que giran a una distancia aproximada de 20 200 km son ejemplos de satélites MEO.

2.4.3 Satélites de Órbita Terrestre Baja (LEO)

Los satélites **LEO** (**Órbita Terrestre Baja**, del inglés *Low-Earth Orbit*) se encuentran a una altitud todavía más baja. Debido a su rápido movimiento, se necesita un gran número de ellos para un sistema completo. Por otro lado, como los satélites están tan cerca de la Tierra, las estaciones terrestres no necesitan mucha potencia y el retardo de viaje redondo es de sólo unos cuantos milisegundos. El costo de lanzamiento es más económico. En esta sección examinaremos dos ejemplos de constelaciones de satélites para el servicio de voz: Iridium y Globalstar.

Durante los primeros 30 años de la era satelital, rara vez se utilizaban los satélites de órbita baja debido a que entran y salen del campo de visión con mucha rapidez. En 1990, Motorola abrió nuevos caminos al presentar una solicitud a la FCC para lanzar 77 satélites de órbita baja para el proyecto **Iridium** (el iridio es el elemento 77). El plan se revisó después para usar sólo 66 satélites, de modo que el proyecto debió haber cambiado su nombre a Dysprosium (elemento 66), pero probablemente eso sonaba muy parecido a una enfermedad. La idea era que, tan pronto como un satélite quedara fuera del campo de visión, otro lo reemplazaría. Esta propuesta desató una gran exaltación entre las demás compañías de comunicaciones. De repente todos querían lanzar una cadena de satélites de órbita baja.

Después de siete años de reunir improvisadamente socios y financiamiento, el servicio de comunicación empezó en noviembre de 1998. Por desgracia, la demanda comercial de teléfonos satelitales grandes y pesados era insignificante debido a que a partir de 1990 la red de telefonía móvil había crecido de manera espectacular. Como consecuencia, Iridium no fue rentable y entró en bancarrota en agosto de 1999, en uno de los fracasos corporativos más espectaculares de la historia. Más adelante un inversionista compró los satélites y otros activos (con valor de $5 mil millones) por $25 millones, en un tipo de venta de garaje extraterrestre. A éste le siguieron casi de inmediato otros proyectos empresariales de satélites.

El servicio Iridium reinició en marzo de 2001 y ha estado creciendo desde entonces. Ofrece servicios de voz, datos, radiolocalización, fax y navegación en tierra, aire y mar, por medio de dispositivos portátiles que se comunican de manera directa con los satélites Iridium. Entre sus clientes se encuentran las industrias marítimas, de aviación y de exploración petrolera, así como las personas que viajan a partes del mundo que carecen de una infraestructura de telecomunicaciones (por ejemplo: desiertos, montañas, el Polo Sur y algunos países del Tercer Mundo).

Los satélites Iridium están posicionados a una altitud de 750 km, en órbitas polares circulares. Están dispuestos en forma de collares de norte a sur con un satélite cada 32 grados de latitud, como se muestra en la figura 2-18. Cada satélite tiene un máximo de 48 celdas (haces puntuales) y una capacidad de 3 840 canales, algunos de los cuales se utilizan para radiolocalización y navegación, mientras que otros se utilizan para datos y voz.

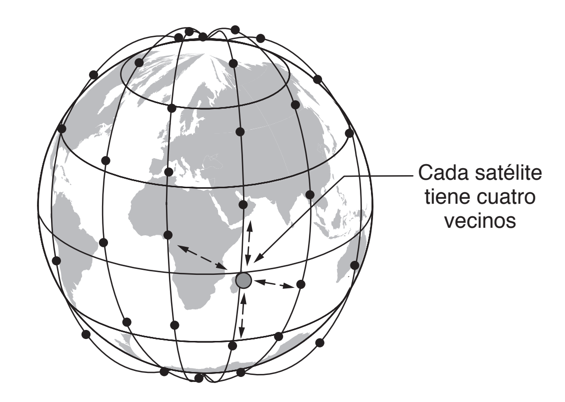


Figura 2-18. Los satélites Iridium forman seis collares alrededor de la Tierra.

Con seis collares de satélites se cubre toda la Tierra, según lo que muestra la figura 2.18. Una propiedad interesante de Iridium es que la comunicación entre clientes distantes se lleva a cabo en el espacio, como se muestra en la figura 2-19(a). En este ejemplo tenemos una persona que llama desde el Polo Norte y se comunica con un satélite que está directamente encima de ella. Cada satélite tiene cuatro vecinos con los que se puede comunicar, dos en el mismo collar (mostrados) y dos en collares adyacentes (no se muestran). Los satélites transmiten la llamada a través de esta rejilla hasta que se envía hacia abajo a la persona que recibe la llamada en el Polo Sur.

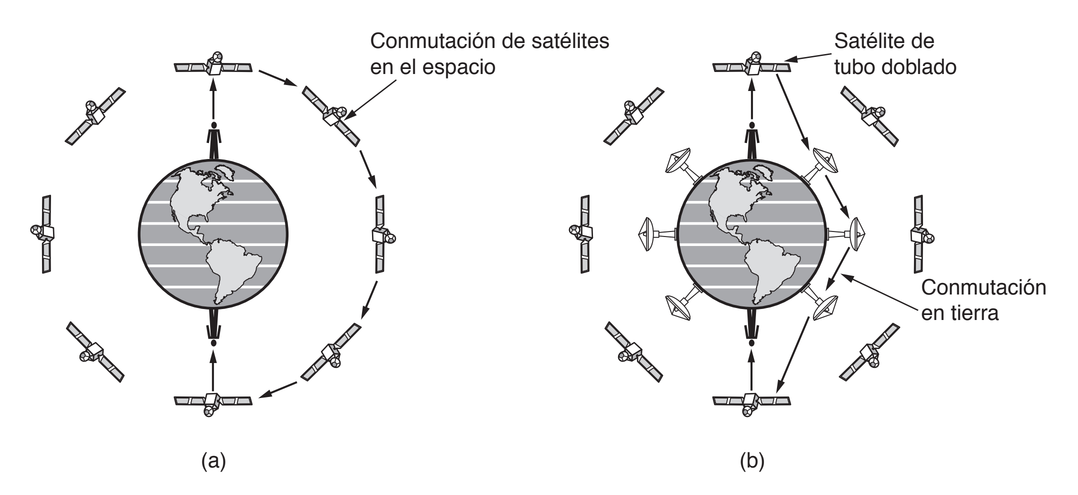


Figura 2-19. (a) Transmisión en el espacio. (b) Transmisión en tierra.

Una alternativa de diseño para Iridium es **Globalstar**, que se basa en 48 satélites LEO pero utiliza un esquema de conmutación distinto al de Iridium. Mientras que Iridium transmite llamadas de un satélite a otro, para lo cual se requiere de un sofisticado equipo de conmutación en los satélites, Globalstar utiliza un diseño tradicional de tubo doblado. La llamada que se origina en el Polo Norte en la figura 2-19(b) se envía de vuelta a la Tierra y es recogida por la gran estación terrestre en el "Taller de Santa". Después la llamada se encamina a través de una red terrestre hacia la estación en tierra más cercana a la persona que va a recibir la llamada y se entrega mediante una conexión de tubo doblado, como se muestra en la imagen. La ventaja de este esquema es que la mayor parte de la complejidad está en tierra, en donde es más fácil de manejar. Además, el uso de antenas grandes en la estación en tierra que pueden enviar una señal potente y recibir una débil significa que se pueden utilizar teléfonos de baja potencia. Después de todo, el teléfono sólo emite unos cuantos miliwatts de potencia, de modo que la señal que llega a la estación en tierra es muy débil, aún después de que el satélite la haya amplificado.

En la actualidad se siguen lanzando satélites a razón de 20 por año, incluyendo los satélites cada vez más grandes que ahora pesan más de 5 000 kilogramos. Pero también hay satélites muy pequeños para las empresas con presupuesto limitado. Para que la investigación espacial fuera más accesible, los académicos de Cal Poly y Stanford se reunieron en 1999 y definieron un estándar para satélites miniatura y un lanzador asociado que reduciría en forma considerable los costos de lanzamiento (Nugent y colaboradores, 2008). Los **CubeSat** son satélites en unidades de cubos de $10 \times 10 \times 10$ cm, cada uno de los cuales pesa cuando mucho 1 kilogramo, y se pueden lanzar por la módica cantidad de $40 000 cada uno. El lanzador vuela como una carga secundaria en las misiones espaciales comerciales. Es en esencia un tubo que almacena hasta tres unidades de satélites CubeSat y utiliza resortes para liberarlos en órbita. Hasta ahora se han lanzado cerca de 20 CubeSat, y hay muchos más en proceso. La mayoría de ellos se comunican con estaciones terrestres en las bandas UHF y VHF.

2.4.4 Comparación de los satélites y la fibra óptica

Una comparación entre la comunicación vía satélite y la comunicación terrestre es algo instructivo. Apenas hace 25 años podríamos argumentar que el futuro de las comunicaciones recaería en los satélites de comunicación. Después de todo, el sistema telefónico había cambiado muy poco en los 100 años anteriores y no mostraba signos de cambio en los siguientes 100. Este movimiento glacial era provocado en gran parte por el entorno regulatorio en el que se exigía a las compañías telefónicas proveer un buen servicio de voz a precios razonables (lo cual cumplían), y a cambio obtenían ganancias garantizadas sobre su inversión. Para las personas que necesitaban transmitir datos, había módems de 1200 bps disponibles. Eso era prácticamente todo lo que se tenía.

La introducción de la competencia en 1984 en Estados Unidos y poco después en Europa cambió todo eso de manera radical. Las compañías telefónicas empezaron a reemplazar sus redes de larga distancia con fibra óptica e introdujeron servicios con ancho de banda alto, como **ADSL** (**Línea asimétrica de suscriptor digital**, del inglés *Asymmetric Digital Subscriber Line)*. También dejaron su antigua práctica de cobrar precios estratosféricos a los usuarios de larga distancia para subsidiar el servicio local. De repente, las conexiones terrestres de fibra óptica se perfilaban como el ganador.

Sin embargo, los satélites de comunicación tienen algunos mercados de nichos importantes que la fibra óptica no maneja (y en ciertos casos porque no puede hacerlo). En primer lugar, cuando es imprescindible un despliegue rápido, los satélites ganan fácilmente. Una respuesta rápida es útil para los sistemas de comunicaciones militares en tiempos de guerra y para la respuesta al desastre en tiempos de paz. Por ejemplo, después del terremoto masivo en Sumatra en diciembre de 2004 y del posterior tsunami, los satélites de comunicaciones pudieron restablecer las comunicaciones con los primeros respondedores en un plazo no mayor de 24 horas. Esta rápida respuesta fue posible debido a que existe un mercado desarrollado de proveedores de servicios de satélite en donde los principales participantes, como Intelsat con más de 50 satélites, pueden rentar su capacidad casi en cualquier parte en donde se requiera. Para los clientes que reciben servicio de las redes satelitales existentes, se puede establecer un sistema VSAT con facilidad y rapidez para proveer un enlace de un megabit/seg a cualquier parte del mundo.

Un segundo nicho es para la comunicación en lugares en donde la infraestructura terrestre está mal desarrollada. En la actualidad muchas personas desean comunicarse desde cualquier parte a donde vayan. Las redes de telefonía móvil cubren esas ubicaciones con buena densidad de población, pero no realizan un trabajo adecuado en otros lugares (por ejemplo, en el mar o en el desierto). En contraste, Iridium provee servicio de voz en cualquier lugar de la Tierra, incluso en el Polo Sur. Además, la instalación de la infraestructura terrestre puede ser costosa, dependiendo del terreno y de los derechos de paso necesarios. Por ejemplo, Indonesia tiene su propio satélite para el tráfico de telefonía nacional. Fue más económico lanzar un satélite que tender miles de cables bajo el mar entre las 13 677 islas en el archipiélago.

En el tercer nicho la difusión es imprescindible. El mensaje que envía un satélite lo pueden recibir miles de estaciones terrestres a la vez. Por esta razón, los satélites se utilizan para distribuir gran parte de la programación de TV a las estaciones locales. Ahora hay un extenso mercado para las difusiones vía satélite de TV y radio digital directamente a los usuarios finales, que cuentan con receptores de satélite en sus hogares y autos. También se pueden difundir otros tipos de contenido. Por ejemplo, tal vez para una organización que transmite un flujo de precios de acciones, bonos o materia prima a miles de distribuidores sea más económico utilizar un sistema de satélite que simular la difusión en la Tierra.

En resumen, parece ser que las comunicaciones dominantes en el futuro serán a través de la fibra óptica terrestre combinada con la radio celular, pero para ciertos usos especializados son mejores los satélites. Sin embargo, hay una advertencia que se aplica a todo esto: la economía. Aunque la fibra óptica ofrece más ancho de banda, es probable que la comunicación terrestre y la comunicación vía satélite puedan competir de manera agresiva en cuanto al precio. Si los avances en la tecnología reducen de manera

radical el costo de desplegar un satélite (por ejemplo, si algún vehículo espacial en el futuro puede lanzar docenas de satélites a la vez) o los satélites de órbita baja presentan avances considerables, no es seguro que la fibra óptica vaya a ganar en todos los mercados.

2.5 MODULACIÓN DIGITAL Y MULTIPLEXIÓN

Ahora que hemos estudiado las propiedades de los canales alámbricos e inalámbricos, nos enfocaremos en el problema de cómo enviar información digital. Los cables y los canales inalámbricos transportan señales analógicas, como el voltaje, la intensidad de la luz o del sonido que varían de forma continua. Para enviar información digital debemos idear señales analógicas que representen bits. Al proceso de realizar la conversión entre los bits y las señales que los representan se le conoce como **modulación digital**.

Empezaremos con esquemas que convierten directamente los bits en una señal. Estos esquemas resultan en una **transmisión en banda base**, en donde la señal ocupa frecuencias desde cero hasta un valor máximo que depende de la tasa de señalización. Este tipo de transmisión es común para los cables. Después consideraremos esquemas que varían la amplitud, fase o frecuencia de una señal portadora para transmitir los bits. Estos esquemas resultan en una **transmisión pasa-banda**, en donde la señal ocupa una banda de frecuencias alrededor de la frecuencia de la señal portadora. Es común para los canales inalámbricos y ópticos, en donde las señales deben residir en una banda de frecuencia dada.

A menudo los canales se comparten entre varias señales. Después de todo, es mucho más conveniente utilizar un solo cable para transportar varias señales que instalar un cable para cada señal. A este tipo de compartición se le denomina **multiplexión** y se puede lograr de varias formas. Presentaremos los métodos para la multiplexión por división de tiempo, de frecuencia y de código.

Las técnicas de modulación y multiplexión que describiremos en esta sección son muy usados en los cables, la fibra óptica, los canales inalámbricos terrestres y los canales de satélite. En las siguientes secciones analizaremos ejemplos de redes para verlos en acción.

2.5.1 Transmisión en banda base

La forma más simple de modulación digital es utilizar un voltaje positivo para representar un 1 y un voltaje negativo para representar un 0. Para una fibra óptica, la presencia de luz podría representar un 1 y la ausencia de luz podría representar un 0. Este esquema se denomina **NRZ** (**No Retorno a Cero**, del inglés *Non-Return-to-Zero*). El nombre extraño es por cuestiones históricas; simplemente significa que la señal sigue a los datos. En la figura 2-20(b) se muestra un ejemplo.

Una vez enviada, la señal NRZ se propaga por el cable. En el otro extremo, el receptor la convierte en bits al muestrear la señal a intervalos de tiempo regulares. Esta señal no se verá exactamente igual que la señal que se envió. El canal y el ruido en el receptor la atenuarán y distorsionarán. Para decodificar los bits, el receptor asocia las muestras de la señal con los símbolos más cercanos. Para NRZ, se tomará un voltaje positivo para indicar que se envió un 1 y un voltaje negativo para indicar que se envió un 0.

El esquema NRZ es un buen punto de inicio para nuestros estudios, ya que es simple pero se utiliza pocas veces por sí solo en la práctica. Los esquemas más complejos pueden convertir bits en señales que cumplen mejor con las consideraciones de ingeniería. Estos esquemas se denominan **códigos de línea**. A continuación describiremos códigos de línea que ayudan con la eficiencia del ancho de banda, la recuperación del reloj y el balance de CD.

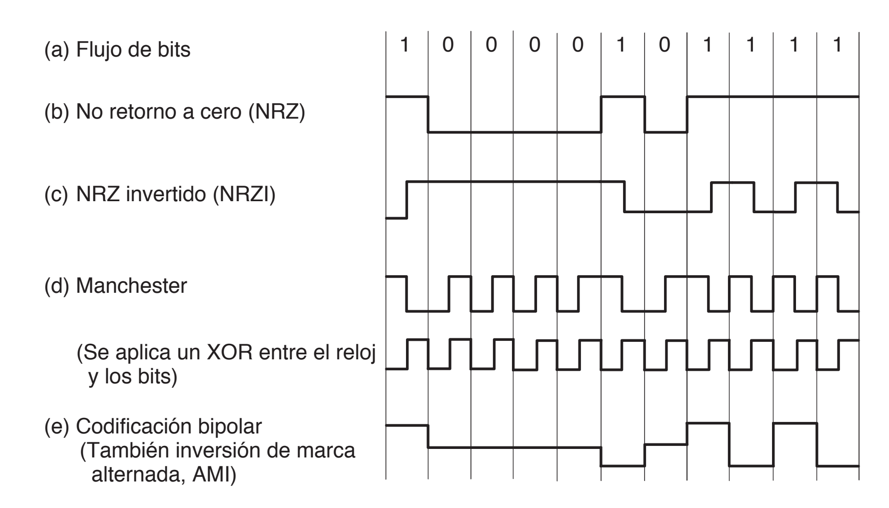


Figura 2-20. Códigos de línea: (a) Bits, (b) NRZ, (c) NRZI, (d) Manchester, (e) Bipolar o AMI.

Eficiencia del ancho de banda

Con NRZ, la señal puede alternar entre los niveles positivo y negativo hasta cada 2 bits (en caso de alternar 1 s y 0 s). Esto significa que necesitamos un ancho de banda de por lo menos *B*/2 Hz cuando la tasa de bits es de *B* bits/seg. Esta relación proviene de la tasa de Nyquist [ecuación (2-2)]. Es un límite fundamental, por lo que no podemos operar el esquema NRZ a una mayor velocidad sin usar más ancho de banda. Por lo general el ancho de banda es un recurso limitado, incluso para los canales con cables. Entre más altas sean las frecuencias de las señales su atenuación es cada vez mayor, lo que las hace menos útiles; además las señales de frecuencias más altas también requieren componentes electrónicos más rápidos.

Una estrategia para utilizar el ancho de banda limitado con más eficiencia es usar más de dos niveles de señalización. Por ejemplo, si utilizamos cuatro voltajes podemos enviar 2 bits a la vez como un solo **símbolo**. Este diseño funcionará siempre y cuando la señal en el receptor sea lo bastante fuerte como para diferenciar los cuatro niveles. La tasa a la que cambia la señal es entonces la mitad de la tasa de bits, por lo que se reduce el ancho de banda necesario.

La tasa a la que cambia la señal se denomina **tasa de símbolo** para diferenciarla de la **tasa de bits**. La tasa de bits es la tasa de símbolo multiplicada por el número de bits por símbolo. Un nombre antiguo para la tasa de símbolo, en especial dentro del contexto de los dispositivos conocidos como módems telefónicos que transmiten datos digitales a través de las líneas telefónicas, es la **tasa de baudios**. En la literatura es frecuente que los términos "tasa de bits" y "tasa de baudios" se usen en forma incorrecta.

Hay que tener en cuenta que el número de niveles de la señal no necesita ser una potencia de dos. A menudo no lo es, ya que algunos de los niveles se utilizan como protección contra errores y para simplificar el diseño del receptor.

Recuperación del reloj

En todos los esquemas que codifican bits en símbolos, el receptor debe saber cuándo termina un símbolo y empieza el siguiente para decodificar los bits en forma correcta. En el esquema NRZ, en donde los símbolos son sólo niveles de voltaje, una larga sucesión de 0 s o 1 s deja la señal sin cambios. Después de un rato es difícil diferenciar unos bits de otros, puesto que 15 ceros se ven muy parecidos a 16 ceros, a menos que usted cuente con un reloj muy exacto.

Los relojes exactos serían útiles para resolver este problema, pero son una solución costosa para un equipo básico. Recuerde que vamos a sincronizar bits en enlaces que operan a muchos megabits/seg, por lo que el reloj tendría que variar menos de una fracción de un microsegundo durante la sucesión más larga permitida. Esto podría ser razonable para los enlaces lentos o mensajes cortos, pero no es una solución general.

Una estrategia es enviar una señal de reloj separada al receptor. Otra línea de reloj no representa mucho para los buses de computadora o los cables cortos en donde hay muchas líneas en paralelo, pero sería un desperdicio para la mayoría de los enlaces de red, ya que si tuviéramos otra línea para enviar una señal, la podríamos usar para enviar datos. Un astuto truco que se usa es mezclar la señal de reloj con la señal de datos, mediante la aplicación de una XOR a ambas señales de manera que no se requiera una línea adicional. En la figura 2-20(d) se muestran los resultados. El reloj realiza una transición en cada tiempo de bit, por lo que opera al doble de la tasa de bits. Al aplicar una XOR con el nivel 0 se produce una transición de nivel bajo a nivel alto, que viene siendo simplemente el reloj. Esta transición es un 0 lógico. Cuando aplica una XOR con el nivel 1, se invierte y produce una transición de nivel alto a nivel bajo. Esta transición es un 1 lógico. Este esquema se llama codificación **Manchester** y se utilizaba en la Ethernet clásica.

La desventaja de la codificación Manchester es que requiere el doble de ancho de banda que NRZ debido al reloj, y hemos aprendido que el ancho de banda es muy importante. Una estrategia distinta se basa en la idea de que deberíamos codificar los datos para asegurar que haya suficientes transiciones en la señal. Consideremos que NRZ tendrá problemas de recuperación del reloj sólo para largas sucesiones de 0s y 1s. Si hay transiciones frecuentes, será fácil para el receptor permanecer sincronizado con el flujo entrante de símbolos.

Para dar un paso en la dirección correcta, podemos simplificar la situación al codificar un 1 como una transición y un 0 como una no transición, o viceversa. A esta codificación se le conoce como **NRZI** (**No Retorno a Cero Invertido**, del inglés *Non-Return-to-Zero Inverted*), un giro sobre el NRZ. En la figura 2-20(c) se muestra un ejemplo. El popular estándar **USB** (**Bus Serie Universal**, del inglés *Universal Serial Bus*) para conectar periféricos de computadora utiliza NZRI. Con él, las largas sucesiones de 1s no provocan problemas.

Desde luego que las largas sucesiones de 0s siguen provocando un problema que debemos corregir. Si fuéramos una compañía telefónica, tal vez sólo tendríamos que requerir que el emisor no transmitiera demasiados 0s. Las primeras líneas telefónicas digitales en Estados Unidos, conocidas como **líneas T1**, de hecho requerían que se enviaran como máximo 15 0s consecutivos para funcionar de forma correcta. Para corregir de verdad este problema, podemos descomponer las sucesiones de 0s y asociar pequeños grupos de bits, para transmitirlos de forma que los grupos con 0s sucesivos se asocien con patrones ligeramente más largos que no tengan muchos 0s consecutivos.

Hay un código muy conocido para hacer esto, el cual se llama **4B/5B**. Aquí se asocian grupos de 4 bits a un patrón de 5 bits con una tabla de traducción fija. Los patrones de 5 bits se elijen de tal forma que nunca haya una sucesión de más de tres 0s consecutivos. La asociación se muestra en la figura 2-21. Este esquema agrega un 25% de sobrecarga, lo cual es mejor que la sobrecarga de 100% de la codificación Manchester. Como hay 16 combinaciones de entrada y 32 de salida, algunas de las combinaciones de salida no se utilizan. Haciendo a un lado las combinaciones con demasiados 0s sucesivos, aún quedan algunos códigos pendientes. Como bono adicional, podemos usar estos códigos sin datos para representar señales de control de la capa física. Por ejemplo, en algunos casos el patrón “11111” representa una línea inactiva y “11000” representa el inicio de una trama.

Una propuesta alternativa es la aleatorización, o *scrambling*, que consiste en hacer que los datos parezcan aleatorios. En este caso es muy probable que haya transiciones frecuentes. La función del *aleatorizador* o *scrambler* es aplicar una XOR entre los datos y una secuencia seudoaleatoria antes de transmitirlos. Esta mezcla hará que los datos sean tan aleatorios como la secuencia seudoaleatoria (suponiendo

Datos (4B)	Palabra de código (5B)	Datos (4B)	Palabra de código (5B)
0000	11110	1000	10010
0001	01001	1001	10011
0010	10100	1010	10110
0011	10101	1011	10111
0100	01010	1100	11010
0101	01011	1101	11011
0110	01110	1110	11100
0111	01111	1111	11101

Figura 2-21. Asociaciones del esquema 4B/5B.

que sean independientes de la secuencia seudoaleatoria). Después el receptor aplica una XOR a los bits entrantes con la misma secuencia seudoaleatoria para recuperar los datos reales. Para que esto sea práctico, la secuencia seudoaleatoria debe ser fácil de crear. Por lo general se proporciona como la semilla para un generador simple de números aleatorios.

La aleatorización es atractiva, porque no añade sobrecarga en el ancho de banda ni en el tiempo. De hecho, a menudo ayuda a acondicionar la señal de manera que no tenga su energía en los componentes de frecuencia dominantes (producidos por los patrones de datos repetitivos) que podrían irradiar interferencia electromagnética. La aleatorización es útil debido a que las señales aleatorias tienden a ser "blancas" o tienen su energía dispersa a través de los componentes de frecuencia.

Sin embargo, la aleatorización no garantiza que no habrá sucesiones largas. Es posible tener mala suerte en algunas ocasiones. Si los datos son iguales que la secuencia seudoaleatoria, al aplicar una XOR todos los bits se convertirán en 0s. Por lo general este resultado no ocurre con una secuencia seudoaleatoria larga que sea difícil de predecir. No obstante, con una secuencia corta o predecible podría darse el caso de que usuarios maliciosos enviaran patrones de bits que provocaran largas sucesiones de 0s después de la aleatorización y causarán fallas en los enlaces. Las primeras versiones de los estándares para enviar paquetes IP a través de enlaces SONET en el sistema telefónico tenían este defecto (Malis y Simpson, 1999). Los usuarios podían enviar ciertos "paquetes asesinos" que garantizaban provocar problemas.

Señales balanceadas

Las señales que tienen la misma cantidad de voltaje positivo y negativo, incluso durante periodos cortos, se conocen como **señales balanceadas**. Su promedio es cero, lo cual significa que no tienen componente eléctrico de CD. La falta de un componente de CD es una ventaja, ya que algunos canales (como el cable coaxial o las líneas con transformadores) atenúan de manera considerable un componente de CD debido a sus propiedades físicas. Además, un método para conectar el receptor al canal, conocido como **acoplamiento capacitivo**, sólo pasa la porción de CA de la señal. En cualquier caso, si enviamos una señal cuyo promedio no sea cero desperdiciaremos energía, puesto que se filtrará el componente de CD.

El balanceo ayuda a proveer transiciones para la recuperación del reloj, ya que hay una mezcla de voltajes positivos y negativos. Además proporciona una forma simple de calibrar los receptores, debido a que se puede medir el promedio de la señal y usarlo como un umbral de decisión para decodificar los símbolos. Con las señales no balanceadas, el promedio puede variar del verdadero nivel de decisión (por ejemplo, debido a una densidad de 1s), lo cual provocaría que se decodificaran más símbolos con errores.

Una manera simple de construir un código balanceado es mediante el uso de dos niveles de voltaje para representar un 1 lógico (por decir, +1 V o −1 V), en donde 0 V representan un cero lógico. Para enviar un 1, el transmisor alterna entre los niveles de +1 V y −1 V de manera que siempre se promedien y eliminen. A este esquema se le llama **codificación bipolar**. En las redes telefónicas se llama **AMI** (**Inversión de Marca Alternada**, del inglés *Alternate Mark Inversion*), con base en la antigua terminología en donde a un 1 se le llama "marca" y a un 0 se le llama "espacio". En la figura 2-20(e) se muestra un ejemplo.

La codificación bipolar agrega un nivel de voltaje para lograr un balance. También podemos usar una asociación como 4B/5B para lograr un balance (así como transiciones para la recuperación del reloj). El código de línea **8B/10B** es un ejemplo de este tipo de código balanceado, en el cual se asocian 8 bits de entrada a 10 bits de salida, por lo cual tiene una eficiencia de 80%, justo igual que el código de línea 4B/5B. Los 8 bits se dividen en un grupo de 5 bits (el cual se asocia a 6 bits) y un grupo de 3 bits (que se asocia a 4 bits). Después se concatenan los símbolos de 6 y 4 bits. En cada grupo se pueden asociar ciertos patrones de entrada a los patrones de salida balanceados que tengan el mismo número de 0s y 1s. Por ejemplo, "001" se asocia con "1001", el cual está balanceado. Pero no hay suficientes combinaciones para que todos los patrones de salida estén balanceados. En estos casos, cada patrón de entrada se asocia a dos patrones de salida. Uno tendrá un 1 extra y el otro tendrá un 0 adicional. Por ejemplo, "000" se asocia a "1011" y a su complemento "0100". A medida que los bits de entrada se asocian a los bits de salida, el codificador recuerda la **disparidad** del símbolo anterior. La disparidad es el número total de 0s o 1s por los que la señal está desbalanceada. Después, el codificador selecciona un patrón de salida o su patrón alterno para reducir la disparidad. Con el código 8B/10B, la disparidad será cuando mucho de 2 bits. Así, la señal nunca estará lejos de ser balanceada. Además, nunca habrá más de cinco 1s o 0s consecutivos para ayudar con la recuperación del reloj.

2.5.2 Transmisión pasa-banda

A menudo es conveniente usar un rango de frecuencias que no empiece en cero para enviar información a través de un canal. En los canales inalámbricos no es práctico enviar señales de muy baja frecuencia, ya que el tamaño de la antena necesita ser de una fracción de la longitud de onda de la señal, por lo que llega a ser grande. En cualquier caso, por lo general la elección de frecuencias se dicta con base en las restricciones regulatorias y a la necesidad de evitar interferencias. Incluso para los cables, es útil colocar una señal en una banda de frecuencias específica para dejar que coexistan distintos tipos de señales en el canal. A este tipo de transmisión se le conoce como **transmisión pasa-banda**, debido a que se utiliza una banda arbitraria de frecuencias para pasar la señal.

Por fortuna, los resultados fundamentales que obtuvimos antes en este capítulo están en términos de ancho de banda, o la anchura de la banda de frecuencias. Los valores absolutos de la frecuencia no importan en cuanto a la capacidad. Esto significa que podemos tomar una señal de **banda base** que ocupe de 0 a B Hz y desplazarla para que ocupe una **banda de paso** de S a $S+B$ Hz sin cambiar la cantidad de información que puede transportar, aun cuando la señal se vea diferente. Para procesar una señal en el receptor, la podemos desplazar de vuelta a la banda base, en donde es más conveniente detectar símbolos.

Para lograr la modulación digital mediante la transmisión pasa-banda, se regula o modula una señal portadora que se sitúa en la banda de paso. Podemos modular la amplitud, frecuencia o fase de la señal portadora. Cada uno de estos métodos tiene su correspondiente nombre. En la **ASK** (**Modulación por Desplazamiento de Amplitud**, del inglés *Amplitude Shift Keying*) se utilizan dos amplitudes distintas para representar el 0 y 1. En la figura 2-22(b) se muestra un ejemplo con un nivel distinto de cero y un nivel 0. Se pueden usar más de dos niveles para representar más símbolos. De manera similar, en la **FSK** (**Modulación por Desplazamiento de Frecuencia**, del inglés *Frequency Shift Keying*) se utilizan dos o más tonos distintos. El ejemplo en la figura 2-22(c) utiliza sólo dos frecuencias. En la forma más simple de **PSK** (**Modulación**

por Desplazamiento de Fase, del inglés *Phase Shift Keying*), la onda portadora se desplaza de manera sistemática 0 o 180 grados en cada periodo de símbolo. Como hay dos fases, se llama **BPSK** (**Modulación por Desplazamiento de Fase Binaria**, del inglés *Binary Phase Shift Keying*). Aquí, la palabra "binaria" se refiere a los dos símbolos, no que los símbolos representan 2 bits. En la figura 2-22(d) se muestra un ejemplo. Un esquema más conveniente en el que se utiliza el ancho de banda del canal con más eficiencia es el que utiliza cuatro desplazamientos (por ejemplo: 45, 135, 225 o 315 grados) para transmitir 2 bits de información por símbolo. Esta versión se llama **QPSK** (**Modulación por Desplazamiento de Fase en Cuadratura**, del inglés *Quadrature Phase Shift Keying*).

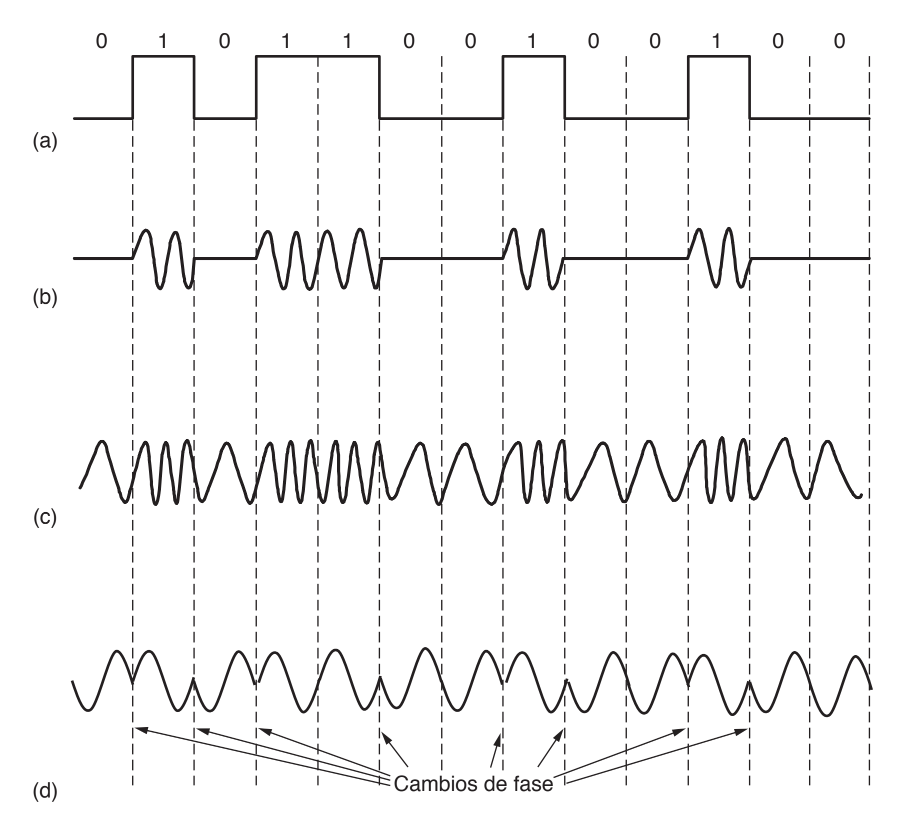


Figura 2-22. (a) Una señal binaria. (b) Modulación por desplazamiento de amplitud. (c) Modulación por desplazamiento de frecuencia. (d) Modulación por desplazamiento de fase.

Podemos combinar estos esquemas y usar más niveles para transmitir más bits por símbolo. Sólo se puede modular la frecuencia o la fase a la vez, ya que están relacionadas; la frecuencia es la tasa de cambio de la fase a través del tiempo. Por lo común, la amplitud y la fase se modulan en combinación. En la figura 2-23 se muestran tres ejemplos. En cada ejemplo, los puntos proporcionan las combinaciones legales de amplitud y fase de cada símbolo. En la figura 2-23(a) podemos ver puntos equidistantes a 45, 135, 225 y 315 grados. La fase de un punto se indica mediante el ángulo que hace una línea (que va desde el punto hasta el origen) con el eje x positivo. La amplitud de un punto es la distancia a partir del origen. Esta figura es una representación de QPSK.

A este tipo de diagrama se le conoce como **diagrama de constelación**. En la figura 2-23(b) podemos ver un esquema de modulación con una constelación más densa. Se utilizan 16 combinaciones de ampli-

tudes y fases, por lo que el esquema de modulación se puede usar para transmitir 4 bits por símbolo. Se denomina **QAM-16**, en donde QAM significa **Modulación de Amplitud en Cuadratura** (*en inglés Quadrature Amplitude Modulatian*). La figura 2-23(c) es un esquema de modulación todavía más denso con 64 combinaciones distintas, por lo que se pueden transmitir 6 bits por símbolo. Se denomina **QAM-64**. También se utilizan esquemas QAM más altos. Como podría sospechar de estas constelaciones, es más fácil construir componentes electrónicos para producir símbolos como una combinación de valores en cada eje, que como una combinación de valores de amplitud y fase. Ésta es la razón por la cual los patrones se ven como cuadros en vez de círculos concéntricos.

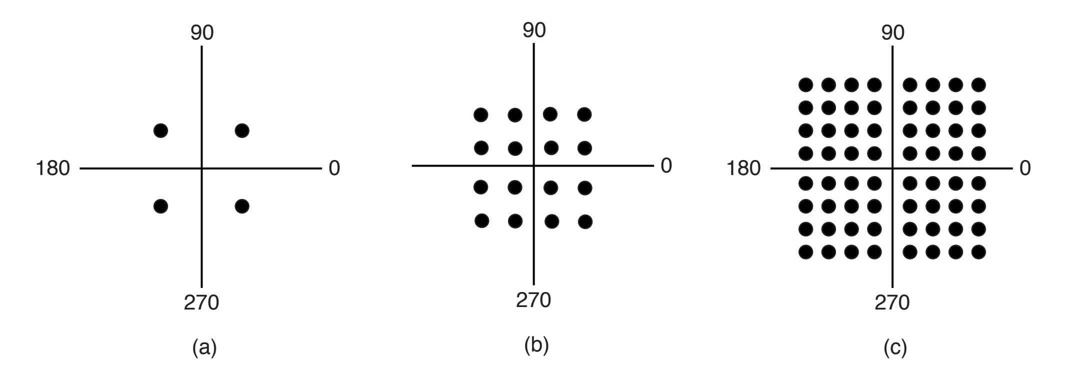


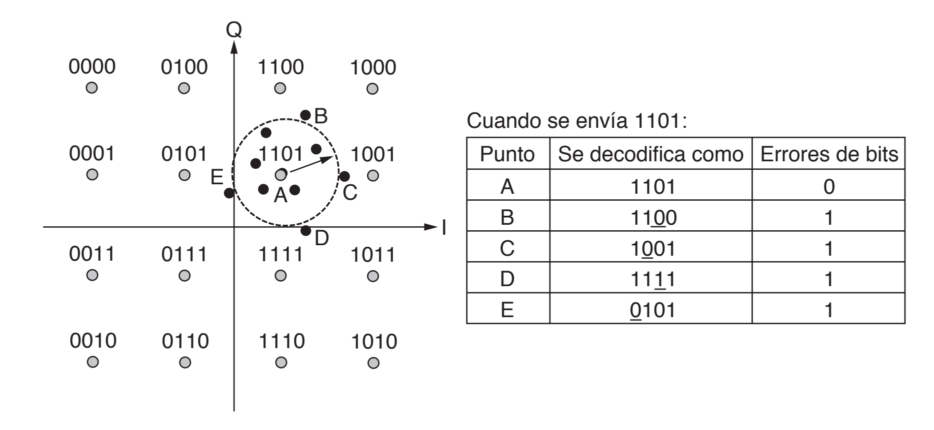
Figura 2-23. (a) QPSK. (b) QAM-16. (c) QAM-64.

Las constelaciones que hemos visto hasta ahora no muestran cómo se asignan los bits a los símbolos. Es importante considerar que al hacer la asignación una pequeña ráfaga de ruido en el receptor no provoque muchos errores de bits. Esto podría ocurrir si asignáramos valores de bits consecutivos a símbolos adyacentes. Con el QAM-16 por ejemplo, si un símbolo representara 0111 y el símbolo adyacente representara 1000, y si el receptor eligiera por error el símbolo adyacente todos los bits estarían incorrectos. Una mejor solución es asociar bits con símbolos de manera que los símbolos adyacentes sólo difieran en 1 posición de bit. A esta asociación se le conoce como **código Gray**. La figura 2-24 muestra una constelación QAM-16 que se ha codificado mediante el código Gray. Ahora, si el receptor decodifica un símbolo por error, sólo cometerá un error de un solo bit en el caso esperado en que el símbolo decodificado esté cerca del símbolo transmitido.

2.5.3 Multiplexión por división de frecuencia

Los esquemas de modulación que hemos visto nos permiten enviar una señal para transmitir bits a través de un enlace alámbrico o inalámbrico. Sin embargo, la economía de escala desempeña un importante papel en cuanto a la forma en que utilizamos las redes. En esencia, es igual de costoso instalar y mantener una línea de transmisión con un alto ancho de banda que una línea con un bajo ancho de banda entre dos oficinas distintas (es decir, los costos provienen de tener que cavar la zanja y no del tipo de cable o fibra óptica que se va a instalar). Por ende, se han desarrollado esquemas de multiplexión para compartir líneas entre muchas señales.

FDM (**Multiplexión por División de Frecuencia**, del inglés *Frecuency Division Multiplexing*) aprovecha la ventaja de la transmisión pasa-banda para compartir un canal. Divide el espectro en bandas de frecuencia, en donde cada usuario tiene posesión exclusiva de cierta banda en la que puede enviar su señal. La difusión de radio AM ilustra el uso del FDM. El espectro asignado es alrededor de 1 MHz, aproximadamente de 500 a 1500 KHz. Las distintas frecuencias se asignan a distintos canales lógicos



Punto	Se decodifica como	Errores de bits
A	1101	0
B	1100	1
C	1001	1
D	1111	1
E	0101	1

Figura 2-24. QAM-16 con código Gray.

(estaciones), cada uno de los cuales opera en una parte del espectro y la separación entre canales es lo bastante grande como para evitar interferencias.

Para un ejemplo más detallado, en la figura 2-25 mostramos tres canales telefónicos de calidad de voz, multiplexados mediante FDM. Los filtros limitan el ancho de banda útil a cerca de 3100 Hz por cada canal de calidad de voz. Cuando se multiplexan muchos canales juntos, se asignan 4 000 Hz por canal. Al exceso se le denomina **banda de guarda**, la cual mantiene los canales bien separados. Primero, los canales de voz se elevan en frecuencia, cada uno en distinto grado. Después se pueden combinar debido a que no hay dos canales que ocupen la misma porción del espectro. Hay que tener en cuenta que, aun cuando hay vacíos entre los canales gracias a las bandas de guarda, existe cierto traslape entre los canales adyacentes. El traslape se debe a que los filtros reales no tienen bordes ideales que sean muy definidos. Esto significa que un pico fuerte en el borde de un canal se detectará en el canal adyacente como ruido no térmico.

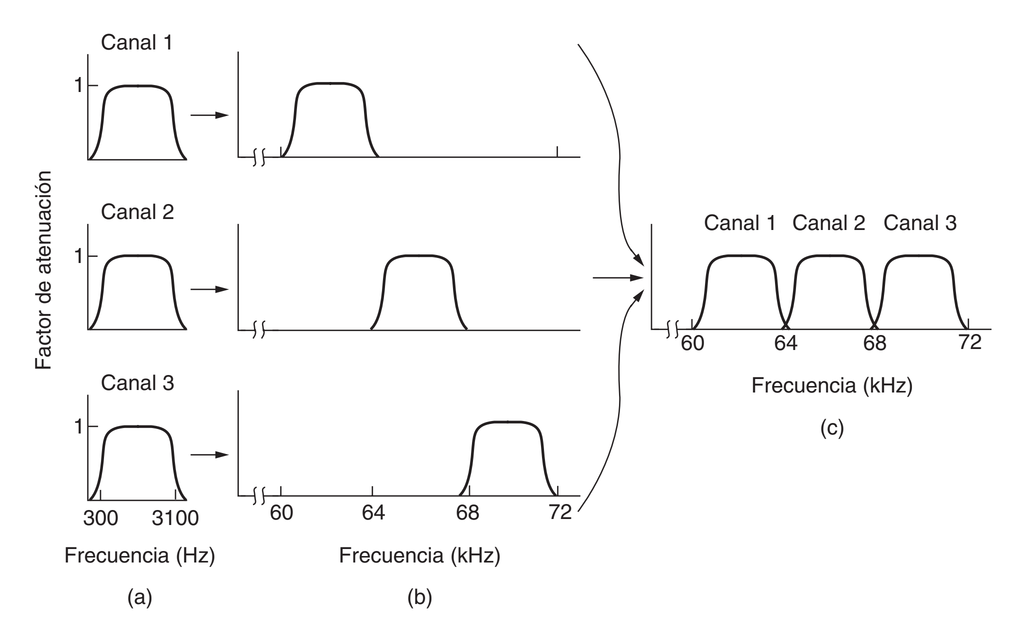


Figura 2-25. Multiplexión por división de frecuencia. (a) Los anchos de banda originales. (b) Los anchos de banda elevados en frecuencia. (c) El canal multiplexado.

Este esquema se ha utilizado para mutiplexar llamadas en el sistema telefónico durante muchos años, pero ahora se prefiere más la multiplexión en el tiempo. Sin embargo, FDM se sigue utilizando en las redes telefónicas, así como en las redes celulares, redes inalámbricas terrestres y redes de satélites con un mayor nivel de granularidad.

Al enviar datos digitales, es posible dividir el espectro de manera eficiente sin usar bandas de guarda. En **OFDM** (**Multiplexión por División de Frecuencia Ortogonal**, del inglés *Orthogonal Frequency Division Multiplexing*), el ancho de banda del canal se divide en muchas subportadoras que envían datos de manera independiente (por ejemplo, mediante QAM). Las subportadoras están empaquetadas estrechamente en el dominio de la frecuencia. Por lo tanto, las señales de cada subportadora se extienden a las subportadoras adyacentes. Pero como podemos ver en la figura 2-26, la respuesta en frecuencia de cada subportadora está diseñada de manera que sea cero en el centro de las subportadoras adyacentes. Por lo tanto, las subportadoras se pueden muestrear en sus frecuencias centrales sin interferencia de sus vecinas. Para que esto funcione, se necesita un tiempo de guarda para repetir una parte de las señales de los símbolos a tiempo, de manera que tengan la respuesta en frecuencia deseada. Sin embargo, esta sobrecarga es mucho menor de lo que se necesita para muchas bandas de guarda.

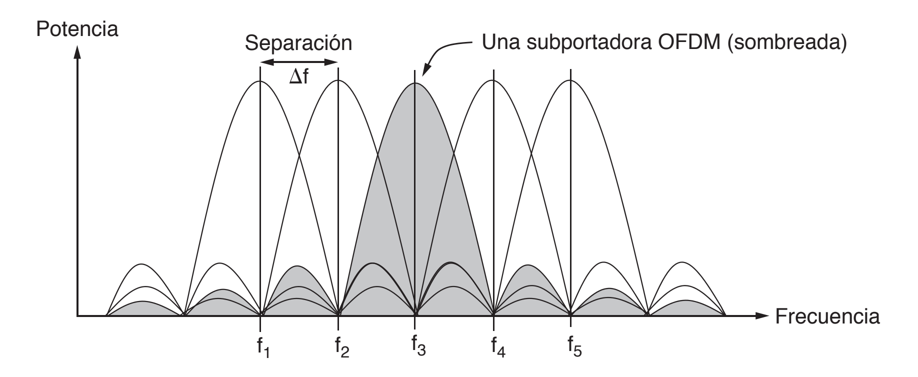


Figura 2-26. Multiplexión por División de Frecuencia Ortogonal (OFDM).

La idea de OFDM ha estado presente por mucho tiempo, pero sólo a partir de esta última década se empezó a adoptar en muchas aplicaciones, después de haberse dado cuenta de que es posible implementar OFDM con eficiencia en términos de una transformada de Fourier de datos digitales sobre todas las subportadoras (en vez de modular por separado cada subportadora). OFDM se utiliza en redes 802.11, de cable y de líneas eléctricas; también hay planes para usarla en los sistemas celulares de cuarta generación. Por lo general, un flujo de alta velocidad de información digital se divide en muchos flujos de baja velocidad que se transmiten en las subportadoras en paralelo. Esta división es valiosa, ya que es más fácil lidiar con las degradaciones del canal a nivel de subportadora; algunas subportadoras pueden estar muy degradadas, por lo que se excluyen a favor de las subportadoras que se reciben bien.

2.5.4 Multiplexión por división de tiempo

TDM (**Multiplexión por División de Tiempo**, del inglés *Time Division Multiplexing*) es una alternativa a FDM. Aquí, los usuarios toman turnos (rotatorios tipo *round-robin*) y cada uno recibe periódicamente todo el ancho de banda durante una pequeña ráfaga de tiempo. En la figura 2-27 se muestra un ejemplo de tres flujos multiplexados mediante TDM. Se toman bits de cada flujo de entrada en una **ranura de tiempo** fija y se envían al flujo agregado. Este flujo opera a una velocidad equivalente a la suma de los flujos

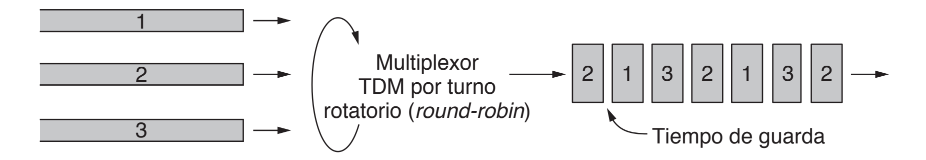


Figura 2-27. Multiplexión por División de Tiempo (TDM).

individuales. Para que esto funcione, los flujos se deben estar sincronizados en tiempo. Se pueden agregar pequeños intervalos de **tiempo de guarda**, los cuales son análogos a una banda de guarda de frecuencia, para tener en cuenta las pequeñas variaciones de sincronización.

El TDM se utiliza mucho como parte de las redes telefónicas y celulares. Para evitar un punto de confusión, dejemos claro que es muy distinto a la **STDM** (**Multiplexión Estadística por División de Tiempo**, del inglés *Statistical Time Division Multiplexing*). El prefijo "estadística" es para indicar que los flujos individuales contribuyen al flujo multiplexado *no* en un itinerario fijo, sino con base en la estadística de su demanda. En sí, STDM es otro nombre para la conmutación de paquetes.

2.5.5 Multiplexión por división de código

Hay un tercer tipo de multiplexión que funciona de una manera muy distinta a FDM y a TDM. **CDM** (**Multiplexión por División de Código**, del inglés *Code Division Multiplexing*) es una forma de comunicación de **espectro disperso** en la que una señal de banda estrecha se dispersa sobre una banda de frecuencia más amplia. Esto puede hacerla más tolerante a la interferencia, al tiempo que permite que varias señales de distintos usuarios compartan la misma banda de frecuencia. Como la multiplexión por división de código se utiliza la mayoría de las veces para este último propósito, se le conoce comúnmente como **CDMA** (**Acceso Múltiple por División de Código**, del inglés *Code Division Multiple Access*).

CDMA permite que cada estación transmita en todo el espectro de frecuencia todo el tiempo. Las múltiples transmisiones simultáneas se separan mediante el uso de la teoría de codificación. Antes de entrar en detalles del algoritmo, consideremos una analogía: una sala de espera en un aeropuerto con muchas parejas conversando. Podemos comparar a TDM con parejas de personas en el cuarto que toman turnos para hablar. FDM es comparable a las parejas de personas que hablan en distintos tonos, algunas en tonos agudos y otras en tonos bajos, de tal forma que cada pareja puede sostener su propia conversación al mismo tiempo, pero de manera independiente a los demás. CDMA se puede comparar con cada pareja de personas que habla a la vez, pero en un lenguaje distinto. La pareja que habla francés sólo se concentra en el francés y rechaza todo lo que no sea francés, pues lo considera ruido. Así, la clave del CDMA es extraer la señal deseada mientras todo lo demás se rechaza como ruido aleatorio. A continuación veremos una descripción algo simplificada de CDMA.

En CDMA, cada tiempo de bit se subdivide en m intervalos cortos llamados **chips**. Por lo general hay 64 o 128 chips por cada bit, pero en el ejemplo que veremos aquí utilizamos 8 chips/bit por cuestión de simpleza. A cada estación se le asigna un código único de m bits, o **secuencia de chip**. Para fines pedagógicos, es conveniente usar una notación bipolar para escribir estos códigos como secuencias de -1 y $+1$. Mostraremos las secuencias de chip entre paréntesis.

Para transmitir un bit 1, una estación envía su secuencia de chip. Para transmitir un bit 0, envía la negación de su secuencia de chip. No se permite ningún otro patrón. Así, para $m = 8$, si se asigna a la estación A la secuencia de chip $(-1\ -1\ -1\ +1\ +1\ -1\ +1\ +1)$, para enviar un bit 1 transmite la secuencia de chip y para enviar un 0 transmite $(+1\ +1\ +1\ -1\ -1\ +1\ -1\ -1)$. En realidad lo que se envía son señales con estos niveles de voltaje, pero es suficiente para nosotros pensar en términos de las secuencias.

La acción de incrementar la cantidad de información a enviar de *b* bits/seg a *mb* chips/seg para cada estación significa que el ancho de banda necesario para CDMA es mayor por un factor de *m* que el ancho de banda necesario para una estación que no utilice CDMA (suponiendo que no haya cambios en las técnicas de modulación o de codificación). Si tenemos una banda de 1 MHz disponible para 100 estaciones, con FDM cada estación tendría 10 kHz y podría enviar a 10 kbps (suponiendo 1 bit por Hz). Con CDMA, cada estación utiliza el 1 MHz completo, por lo que la tasa de chip es de 100 chips por bit para dispersar la tasa de bits de la estación de 10 kbps a través del canal.

En las figuras 2-28(a) y (b) mostramos las secuencias de chip asignadas a cuatro estaciones de ejemplo y las señales que representan. Cada estación tiene su propia secuencia de chip única. Utilizaremos el símbolo **S** para indicar el vector de *m* chips para la estación *S*, y **S** para su negación. Todas las secuencias de chip son **ortogonales** por pares, lo que quiere decir que el producto interno normalizado de dos distintas secuencias de chip cualesquiera, **S** y **T** (lo que se escribe como **S•T**), es 0. Se sabe cómo generar dichas secuencias de chip ortogonales mediante un método conocido como **códigos de Walsh**. En términos matemáticos, la ortogonalidad de las secuencias de chip se puede expresar de la siguiente manera:

$$\mathbf{S} \bullet \mathbf{T} \equiv \frac{1}{m} \sum_{i=1}^{m} S_i T_i = 0 \qquad (2\text{-}5)$$

En español simple, los pares son tan iguales como distintos. Esta propiedad de ortogonalidad demostrará ser imprescindible más adelante. Observe que si **S•T** = 0, entonces **S•T** también es 0. El producto interno normalizado de cualquier secuencia de chip consigo misma es 1:

$$\mathbf{S} \bullet \mathbf{S} = \frac{1}{m} \sum_{i=1}^{m} S_i S_i = \frac{1}{m} \sum_{i=1}^{m} S_i^2 = \frac{1}{m} \sum_{i=1}^{m} (\pm 1)^2 = 1$$

Se deduce esto debido a que cada uno de los *m* términos en el producto interno es 1, por lo que la suma es *m*. Observe además que **S•S** = −1.

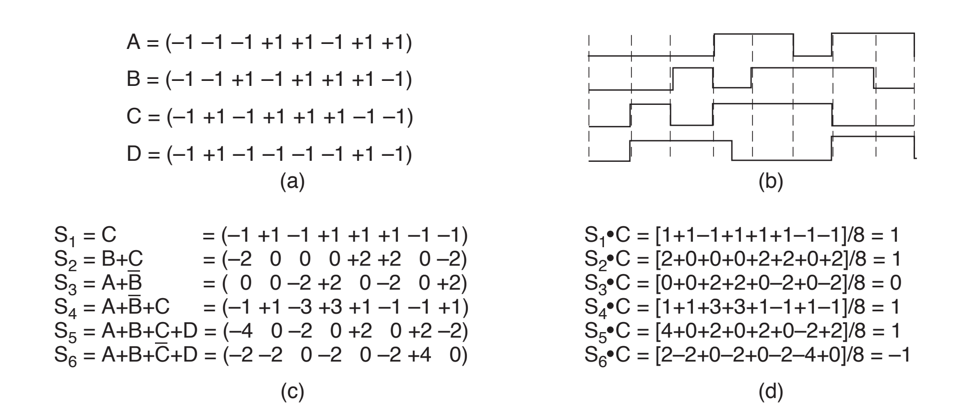


Figura 2-28. (a) Secuencias de chip para cuatro estaciones. (b) Las señales que representan las secuencias. (c) Seis ejemplos de transmisiones. (d) Recuperación de la señal de la estación C.

Durante cada tiempo de bit, una estación puede transmitir un 1 (si envía su secuencia de chip), puede transmitir un 0 (si envía el negativo de su secuencia de chip) o puede permanecer en silencio y no transmitir nada. Por ahora supongamos que todas las estaciones están sincronizadas en el tiempo, por lo que

todas las secuencias de chip empiezan en el mismo instante. Cuando dos o más estaciones transmiten de manera simultánea, sus secuencias bipolares se suman en forma lineal. Por ejemplo, si en un periodo de chip tres estaciones envían +1 y una estación envía −1, se recibirá +2. Podemos considerar esto como señales que se suman a medida que se sobreponen voltajes en el canal: tres estaciones envían +1 V y una estación envía −1 V, de modo que se reciben 2 V. Por ejemplo, en la figura 2-28(c) vemos seis ejemplos de una o más estaciones que transmiten 1 bit al mismo tiempo. En el primer ejemplo, *C* transmite un bit 1, así que sólo recibimos la secuencia de chip de *C*. En el segundo ejemplo, tanto *B* como *C* transmiten bits 1, por lo que obtenemos la suma de sus secuencias de chip bipolares, es decir:

$$(-1\ -1\ +1\ -1\ +1\ +1\ +1\ -1) + (-1\ +1\ -1\ +1\ +1\ +1\ -1\ -1) = (-2\ 0\ 0\ 0\ +2\ +2\ 0\ -2)$$

Para recuperar el flujo de bits de una estación individual, el receptor debe conocer de antemano la secuencia de chip de esa estación. Para llevar a cabo la recuperación, calcula el producto interno normalizado de la secuencia de chip recibida y de la secuencia de chip de la estación cuyo flujo de bits está tratando de recuperar. Si la secuencia de chip recibida es **S** y el receptor trata de escuchar una estación cuya secuencia de chip sea **C**, sólo calcula el producto interno normalizado, **S•C**.

Para ver por qué funciona esto, sólo imagine que dos estaciones *A* y *C* transmiten un bit 1 al mismo tiempo que *B* transmite un bit 0, como se da el caso en el tercer ejemplo. El receptor ve la suma, $\mathbf{S} = \mathbf{A} + \overline{\mathbf{B}} + \mathbf{C}$, y calcula lo siguiente:

$$\mathbf{S} \bullet \mathbf{C} = (\mathbf{A} + \overline{\mathbf{B}} + \mathbf{C}) \bullet \mathbf{C} = \mathbf{A} \bullet \mathbf{C} + \overline{\mathbf{B}} \bullet \mathbf{C} + \mathbf{C} \bullet \mathbf{C} = 0 + 0 + 1 = 1$$

Los primeros dos términos se desvanecen debido a que todos los pares de secuencias de chip se han elegido con cuidado para que sean ortogonales, como se muestra en la ecuación (2-5). Ahora debe quedar claro por qué se debe imponer esta propiedad en las secuencias de chip.

Para que el proceso de decodificación sea más concreto, en la figura 2-28(d) mostramos seis ejemplos. Suponga que el receptor está interesado en extraer el bit enviado por la estación *C* de cada una de las seis señales S_1 a S_6. Para calcular el bit, suma los productos por parejas de la **S** recibida y el vector ***C*** de la figura 2-28(a), y después toma 1/8 del resultado (ya que $m = 8$ en este caso). Los ejemplos incluyen casos en donde *C* está en silencio, envía un bit 1 y envía un bit 0, por separado y en combinación con otras transmisiones. Como se muestra, se decodifica el bit correcto cada vez. Es justo igual que hablar francés.

En principio, dada la suficiente capacidad de cómputo, el receptor puede escuchar a todas las emisoras a la vez si ejecuta el algoritmo de decodificación para cada una de ellas en paralelo. En la vida real basta señalar que es más fácil decirlo que hacerlo, además de que es conveniente saber qué emisoras podrían estar transmitiendo.

En el sistema CDMA ideal sin ruido que hemos estudiado aquí, la cantidad de estaciones que envían datos en forma concurrente puede ser arbitrariamente grande si utilizamos secuencias de chip más largas. Para 2^n estaciones, los códigos de Walsh pueden proveer 2^n secuencias de chip ortogonales de longitud 2^n. No obstante, una limitación considerable es que hemos supuesto que todos los chips están sincronizados en el tiempo en el receptor. Esta sincronización ni siquiera está cerca de ser verdad en algunas aplicaciones, como las redes celulares (en donde se empezó a implementar CDMA en muchos casos desde la década de 1990). Esto conlleva a distintos diseños. Más adelante retomaremos este tema y describiremos la diferencia entre el CDMA asincrónico y el CDMA sincrónico.

Al igual que en las redes celulares, CDMA se utiliza en las redes de satélites y de cable. En esta breve introducción pasamos por alto muchos factores que complicarían el tema. Los ingenieros que deseen obtener una comprensión más detallada de CDMA pueden consultar a Viterbi (1995), y también a Lee y Miller (1998). Sin embargo, estas referencias requieren que el lector tenga un poco de experiencia con la ingeniería de comunicaciones.

2.6 LA RED TELEFÓNICA PÚBLICA CONMUTADA

Cuando una compañía u organización cuenta con dos computadoras que se ubican una cerca de la otra y necesitan comunicarse, con frecuencia lo más fácil es tender un cable entre ellas. Las redes LAN funcionan de esta manera. Sin embargo, cuando las distancias son grandes o hay muchas computadoras, o cuando los cables tienen que pasar por un camino público u otra vía pública, los costos de tender cables privados son por lo general prohibitivos. Además, en casi cualquier país del mundo también es ilegal instalar líneas de transmisión privadas a través (o debajo) de una propiedad pública. Por lo tanto, los diseñadores de redes deben depender de las instalaciones de telecomunicaciones existentes.

Estas instalaciones, en especial la **PSTN** (**Red Telefónica Pública Conmutada**, del inglés *Public Switched Telephone Network*), por lo general se diseñaron hace muchos años con un objetivo completamente distinto en mente: transmitir la voz humana en una forma más o menos reconocible. Su adaptabilidad para usarse en la comunicación de computadora a computadora con frecuencia es marginal en el mejor de los casos. Para ver el tamaño del problema, considere que un cable común y económico tendido entre dos computadoras puede transferir datos a 1 Gbps o más. En contraste, una línea ADSL común, la ultrarrápida alternativa al módem telefónico, opera a una velocidad aproximada de 1 Mbps. La diferencia entre las dos es como viajar en un avión y dar un tranquilo paseo a pie.

Sin embargo, el sistema telefónico está muy entrelazado con las redes de computadoras (de área amplia), por lo que vale la pena dedicar algo de tiempo para estudiarlo con detalle. El factor limitante para fines de interconexión resulta ser la "última milla" a través de la cual se conectan los clientes, no las troncales y conmutadores dentro de la red telefónica. Esta situación está cambiando debido a la extensión gradual de la fibra óptica y la tecnología digital al borde de la red, pero llevará algo de tiempo y dinero. Durante la larga espera, los diseñadores de sistemas de computadoras acostumbrados a trabajar con sistemas que ofrecen un rendimiento por lo menos tres veces mayor, han dedicado mucho tiempo y esfuerzo para averiguar cómo usar la red telefónica en forma eficiente.

En las siguientes secciones describiremos el sistema telefónico y mostraremos cómo funciona. Para obtener información adicional sobre los aspectos internos del sistema telefónico, consulte a Bellamy (2000).

2.6.1 Estructura del sistema telefónico

Poco después de que Alexander Graham Bell patentara el teléfono en 1876 (sólo unas pocas horas antes que su rival, Elisha Gray), hubo una enorme demanda por su nuevo invento. El mercado inicial era para la venta de teléfonos, los cuales se vendían en pares. Al cliente le correspondía tender un cable entre los dos teléfonos. Si el propietario de un teléfono quería hablar a otros n propietarios de teléfonos, tenía que tender cables separados a cada una de las n casas. En menos de un año las ciudades estaban cubiertas de cables que pasaban sobre casas y árboles en un salvaje embrollo. Fue inmediatamente obvio que el modelo de conectar cada teléfono a cada uno de los otros teléfonos, como se muestra en la figura 2-29(a), no iba a funcionar.

Para su fortuna, Bell había previsto este problema y formó la compañía telefónica Bell, la cual abrió su primera oficina de conmutación (en New Haven, Connecticut) en 1878. La compañía tendía un cable hasta la casa u oficina de cada cliente. Para hacer una llamada, el cliente debía dar vueltas a una manivela en el teléfono para producir un sonido de timbre en la oficina de la compañía telefónica y atraer la atención de una operadora, quien a su vez conectaba en forma manual a la persona que llamaba con la persona que iba a recibir la llamada mediante un cable de puenteo. En la figura 2-29(b) se muestra el modelo de una oficina de conmutación.

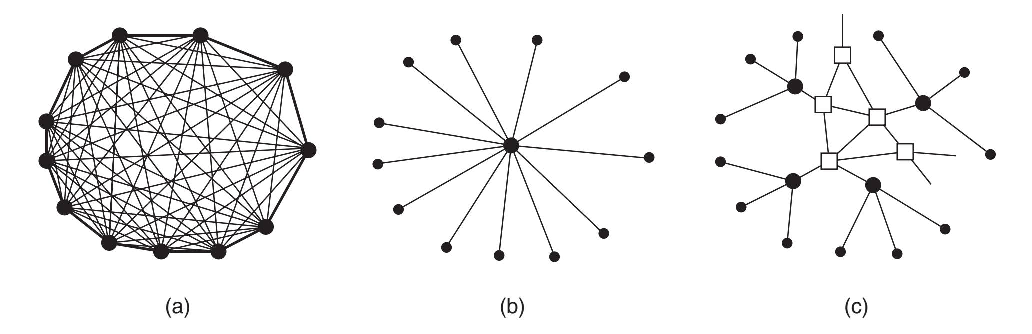


Figura 2-29. (a) Red completamente interconectada. (b) Conmutador centralizado. (c) Jerarquía de dos niveles.

Muy pronto surgieron por todas partes oficinas de conmutación de Bell System y la gente quería hacer llamadas de larga distancia entre ciudades, de modo que el Bell System empezó a conectar las oficinas de conmutación. Pronto reapareció el problema original: conectar cada oficina de conmutación con todas las demás mediante un cable entre ellas pronto dejó de ser práctico, por lo que se inventaron las oficinas de conmutación de segundo nivel, como se muestra en la figura 2-29(c). Con el tiempo, la jerarquía aumentó a cinco niveles.

Para 1890, las tres principales partes del sistema telefónico estaban en operación: las oficinas de conmutación, los cables entre los clientes y las oficinas de conmutación (a estas alturas eran pares trenzados balanceados y aislados, en vez de cables abiertos con retorno a tierra), y las conexiones de larga distancia entre las oficinas de conmutación. Si desea leer una historia técnica corta del sistema telefónico, consulte Hawley (1991).

Aunque desde entonces se han realizado mejoras en las tres áreas, el modelo básico del Bell System ha permanecido en esencia intacto durante más de 100 años. La siguiente descripción está muy simplificada, pero logra transmitir la idea esencial. Cada teléfono tiene dos cables de cobre que salen de él y que van directamente a la **oficina final** más cercana de la compañía telefónica (también se conoce como **oficina central local**). Por lo general la distancia es de 1 a 10 km, siendo menor en las ciudades que en las áreas rurales. Tan sólo en Estados Unidos hay cerca de 22 000 oficinas finales. Las conexiones de dos cables entre el teléfono de cada suscriptor y la oficina central se conocen en el negocio como **lazo local**. Si los lazos locales del mundo se estiraran y unieran por los extremos, se extenderían hasta la Luna y regresarían 1000 veces.

En cierto momento, el 80% del valor del capital de AT&T fue el cobre en los lazos locales. En efecto, AT&T era en ese entonces la mina de cobre más grande del mundo. Por fortuna este hecho no era muy conocido en la comunidad inversionista. De haberse sabido, algún pirata corporativo podría haber comprado AT&T para cancelar todo el servicio telefónico en Estados Unidos, extraer todos los cables y venderlos a algún refinador de cobre para obtener una retribución rápida.

Si un suscriptor conectado a una oficina final determinada llama a otro suscriptor conectado a la misma oficina final, el mecanismo de conmutación dentro de la oficina establece una conexión eléctrica directa entre los dos lazos locales. Esta conexión permanece intacta mientras dure la llamada.

Si el teléfono al que se llama está conectado a otra oficina final, hay que usar un procedimiento distinto. Cada oficina final tiene varias líneas salientes a uno o más centros de conmutación cercanos, llamados **oficinas interurbanas** (o si están dentro de la misma área local, **oficinas en tándem**). Estas líneas se llaman **troncales de conexión interurbanas**. El número de los distintos tipos de centros de conmutación y su topología varía de un país a otro, dependiendo de la densidad telefónica de cada país.

Si sucede que las oficinas finales del que hace y del que recibe la llamada tienen una troncal de conexión interurbana a la misma oficina interurbana (algo muy probable si están lo bastante cerca), se puede establecer la conexión dentro de la oficina interurbana. En la figura 2-29(c) se muestra una red telefónica que consiste sólo de teléfonos (los pequeños puntos), oficinas finales (los puntos grandes) y oficinas interurbanas (los cuadros).

Si la persona que llama y la que recibe la llamada no tienen una oficina interurbana en común, habrá que establecer una trayectoria entre dos oficinas interurbanas. Las oficinas interurbanas se comunican entre sí mediante **troncales interurbanas** que cuentan con un gran ancho de banda (también se les conoce como **troncales interoficinas**). Antes de la disolución de AT&T en 1984, el sistema telefónico de Estados Unidos usaba un enrutamiento jerárquico para buscar una trayectoria y pasaba a niveles superiores en la jerarquía hasta encontrar una oficina de conmutación en común. Después, esto se reemplazó con un enrutamiento más flexible sin jerarquías. La figura 2-30 muestra cómo se podría enrutar una conexión de larga distancia.

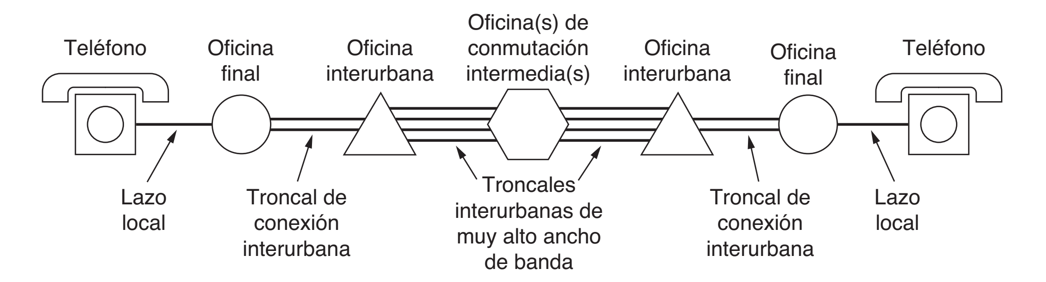


Figura 2-30. Ruta típica de un circuito para una llamada de larga distancia.

Para las telecomunicaciones se utiliza una variedad de medios de transmisión. A diferencia de los edificios modernos de oficinas en donde el cableado es por lo general de categoría 5, los lazos locales a los hogares consisten en su mayoría de pares trenzados categoría 3, y en algunos lugares está empezando a aparecer la fibra óptica. Entre las oficinas de conmutación se utilizan cables coaxiales, microondas y en especial fibra óptica.

En el pasado, la transmisión en todo el sistema telefónico era analógica, en donde la señal de voz en sí se transmitía como un voltaje eléctrico desde el origen hasta el destino. Con la llegada de la fibra óptica, la electrónica digital y las computadoras, ahora todas las troncales y conmutadores son digitales, y el lazo local queda como último elemento de tecnología analógica en el sistema. Es preferible la transmisión digital debido a que no es necesario reproducir con exactitud una forma de onda analógica después de haber pasado por muchos amplificadores en una llamada larga. Basta con distinguir correctamente un 0 de un 1. Esta propiedad hace que la transmisión digital sea más confiable que la analógica. Además es más económica y fácil de mantener.

En resumen, el sistema telefónico está constituido por tres componentes principales:

1. Lazos locales (pares trenzados analógicos que van a los hogares y negocios).
2. Troncales (enlaces digitales de fibra óptica que conectan las oficinas de conmutación).
3. Oficinas de conmutación (en donde las llamadas se pasan de una troncal a otra).

Después de una breve digresión sobre la política de los teléfonos, analizaremos cada uno de estos tres componentes detalladamente. Los lazos locales proveen acceso para todos al sistema completo, por lo que son cruciales. Por desgracia también son el eslabón más débil en el sistema. Para las troncales de largo

alcance, la consideración principal es cómo reunir varias llamadas y enviarlas a través de la misma fibra. Para ello se requiere la multiplexión a través de FDM y TDM. Por último, hay dos formas fundamentalmente distintas de realizar la conmutación; analizaremos ambas.

2.6.2 La política de los teléfonos

Durante las décadas anteriores a 1984, el Bell System proporcionaba tanto el servicio local como el de larga distancia en casi todo el territorio de Estados Unidos. En la década de 1970, el gobierno federal estadounidense se convenció de que era un monopolio ilegal y entabló un juicio para dividirlo. El gobierno ganó, y el 1 de enero de 1984, AT&T se dividió en AT&T Long Lines, 23 compañías **BOC** (**Compañías Operativas de Bell**, del inglés *Bell Operating Companies*) y algunas otras partes pequeñas. Las 23 BOC se agruparon en siete BOC regionales (RBOC) para hacerlas económicamente viables. La naturaleza entera de las telecomunicaciones en Estados Unidos se cambió de la noche a la mañana por orden judicial (*no* por un acto del Congreso).

Las especificaciones exactas del desmantelamiento se describieron en el llamado **MFJ** (**Juicio Final Modificado**, del inglés *Modified Final Judgment*), un contrasentido si alguna vez hubo uno (si el juicio se podía modificar, obviamente no era final). Este suceso condujo a un aumento en la competencia, un mejor servicio y tarifas de larga distancia más bajas para los consumidores y las empresas. Sin embargo, los precios del servicio local aumentaron al eliminar los subsidios cruzados de las llamadas de larga distancia, de modo que el servicio local tuvo que independizarse económicamente. Ahora muchos otros países han introducido la competencia por caminos similares.

Algo de relevancia directa para nuestros estudios es que el nuevo marco de trabajo competitivo provocó que se agregara una característica técnica clave a la arquitectura de la red telefónica. Para dejar en claro quiénes podían actuar y cómo, Estados Unidos se dividió en 164 áreas **LATA** (**Áreas de Acceso Local y de Transporte**, del inglés *Local Access and Transport Areas*). A grandes rasgos, una LATA es casi tan grande como el área cubierta por un código de área. Dentro de cada LATA había una **LEC** (**Portadora de Intercambio Local**, del inglés *Local Exchange Carrier*) con un monopolio del servicio tradicional de telefonía dentro de su área. Las LEC más importantes eran las BOC, aunque algunas LATA contenían una o más de las 1 500 compañías telefónicas independientes que operaban como LEC.

La nueva característica era que todo el tráfico dentro de las LATA se manejaba a través de un tipo distinto de compañía: una **IXC** (**Portadora entre Centrales**, del inglés *IntereXchange Carrier*). En un principio, AT&T Long Lines era la única IXC seria, pero ahora hay competidores bien establecidos como Verizon y Sprint en el negocio de las IXC. Una de las consideraciones durante la disolución fue asegurar que todas las IXC se tratarían con igualdad en términos de calidad de las líneas, tarifas y cantidad de dígitos que sus clientes tendrían que marcar para usarlas. En la figura 2-31 se ilustra la forma en que se maneja esto. Aquí podemos ver tres LATA de ejemplo, cada una con varias oficinas finales. Las LATA 2 y 3 también tienen una pequeña jerarquía con oficinas tándem (oficinas interurbanas inter-LATA).

Cualquier IXC que desee manejar llamadas que se originen en una LATA puede construir allí una oficina de conmutación conocida como **POP** (**Punto de Presencia**, del inglés *Point of Presence*). Se requiere la LEC para conectar cada IXC con cada oficina final, ya sea de manera directa como en las LATA 1 y 3, o de manera indirecta como en la LATA 2. Asimismo, los términos de la conexión (tanto técnicos como financieros) deben ser idénticos para todas las IXC. De esta forma, este requerimiento permite que un suscriptor en la LATA 1 (por ejemplo) seleccione cuál IXC desea usar para llamar a los suscriptores en la LATA 3.

Como parte del MFJ, se prohibió a las IXC ofrecer servicio telefónico local y a las LEC ofrecer servicio telefónico inter-LATA, aunque ambas tenían libertad de participar en cualquier otro negocio, como operar restaurantes de pollos fritos. En 1984 éste era un dictamen bastante claro. Por desgracia la tecnología tiene una forma graciosa de hacer obsoletas las leyes. Ni la televisión por cable ni los teléfonos

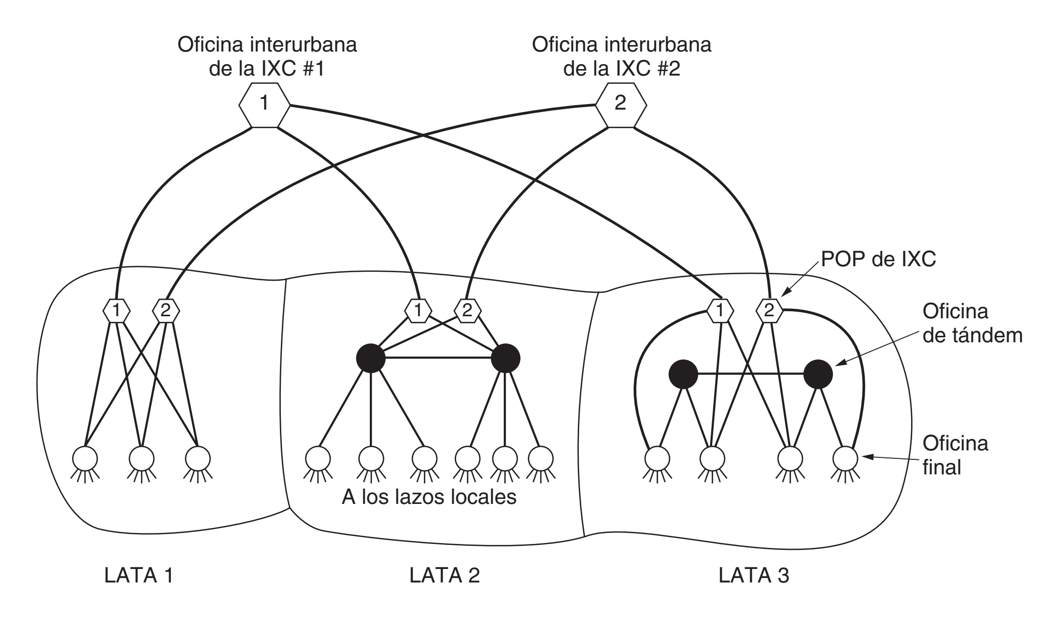


Figura 2-31. La relación entre las LATA, LEC e IXC. Todos los círculos son oficinas de conmutación LEC. Cada hexágono pertenece a la IXC cuyo número contiene.

celulares estaban cubiertos por este acuerdo. Cuando la televisión por cable pasó de ser unidireccional a bidireccional y explotó la popularidad de los teléfonos móviles, tanto las LEC como las IXC empezaron a comprar o a fusionarse con las operadoras de cable y de celulares.

Para 1995 el Congreso vio que tratar de mantener la distinción entre las diversas clases de compañías ya no era sostenible y esbozó una propuesta de ley para preservar la accesibilidad de la competencia pero a la vez permitir a las compañías de TV por cable, las compañías telefónicas locales, las portadoras de larga distancia y las operadoras móviles participar en los negocios de unas y otras. La idea era que cualquier empresa podría entonces ofrecer a sus clientes un solo paquete integrado que incluyera TV por cable, teléfono y servicios de información, y que las distintas compañías compitieran en cuanto a servicio y precio. La propuesta se convirtió en ley en febrero de 1996 como una importante revisión en la regulación de las telecomunicaciones. Como resultado, algunas BOC se convirtieron en IXC y algunas otras compañías (como las operadoras de televisión por cable) empezaron a ofrecer el servicio telefónico local para competir con las LEC.

Una interesante propiedad de la ley de 1996 es el requerimiento de que las LEC deben implementar una **portabilidad para números locales**. Esto significa que un cliente puede cambiar de compañía telefónica local sin tener que obtener un nuevo número telefónico. A ésta le siguió la portabilidad para los números telefónicos móviles (y entre las líneas fijas y móviles) en 2003. Estas cláusulas eliminaron un enorme obstáculo para muchas personas, quienes se vieron más inclinadas a cambiar de LEC. Como resultado, el panorama de las telecomunicaciones en Estados Unidos se volvió mucho más competitivo; otros países han seguido el mismo camino. Con frecuencia los otros países quieren ver cómo funciona este tipo de experimento en Estados Unidos. Si funciona bien, hacen lo mismo; si funciona mal, prueban otra cosa.

2.6.3 El lazo local: módems, ADSL y fibra óptica

Ahora es tiempo de empezar con nuestro estudio detallado sobre cómo funciona el sistema telefónico. Comencemos con la parte que la mayoría de la gente conoce: el lazo local de dos alambres que provie-

ne de una oficina final de la compañía telefónica y llega a los hogares. El lazo local también se conoce comúnmente como la "última milla", aunque la longitud puede ser de hasta varias millas. Ha transportado información analógica por más de 100 años y es probable que siga haciéndolo por otros años más, debido al alto costo de la conversión a digital.

Se han hecho muchos esfuerzos por sacar las redes de datos de los lazos locales de cobre que ya están implementados. Los módems telefónicos envían datos digitales entre computadoras a través del angosto canal que provee la red telefónica para una llamada de voz. Alguna vez fueron muy populares, pero han sido reemplazados por las tecnologías de banda ancha como ADSL, que reutilizan el lazo local para enviar datos digitales desde un cliente hasta la oficina final, en donde se descargan hacia Internet. Tanto los módems como la tecnología ADSL tienen que lidiar con las limitaciones de los viejos lazos locales: un ancho de banda relativamente estrecho, atenuación y distorsión de las señales, y susceptibilidad al ruido eléctrico, como la diafonía.

En algunos lugares se ha modernizado el lazo local mediante la instalación de fibra óptica hasta (o muy cerca de) el hogar. La fibra es la vía del futuro. Estas instalaciones soportan redes de computadoras desde el inicio, y el lazo local tiene un amplio ancho de banda para los servicios de datos. El factor limitante es lo que las personas tienen que pagar, no los aspectos físicos del lazo local.

En esta sección estudiaremos tanto el lazo local antiguo como el nuevo. Veremos los módems telefónicos, ADSL y la fibra hasta el hogar.

Módems telefónicos

Para enviar bits a través del lazo local o de cualquier otro canal físico en sí, hay que convertirlos a señales analógicas que se puedan transmitir por el canal. Para lograr esta conversión se utilizan los métodos de modulación digital que estudiamos en la sección anterior. Al otro extremo del canal, la señal analógica se convierte de vuelta en bits.

Un dispositivo que realiza conversiones entre un flujo de bits digitales y una señal analógica que representa esos bits se llama **módem**, que es una abreviación de "*mo*dulador *dem*odulador". Hay muchas variedades de módems: telefónicos, DSL, de cable, inalámbricos etc. El módem puede estar integrado en la computadora (lo cual es ahora común para los módems telefónicos) o puede ser una caja separada (algo común para los módems DSL y de cable). Lógicamente el módem se inserta entre la computadora (digital) y el sistema telefónico (analógico), como podemos ver en la figura 2-32.

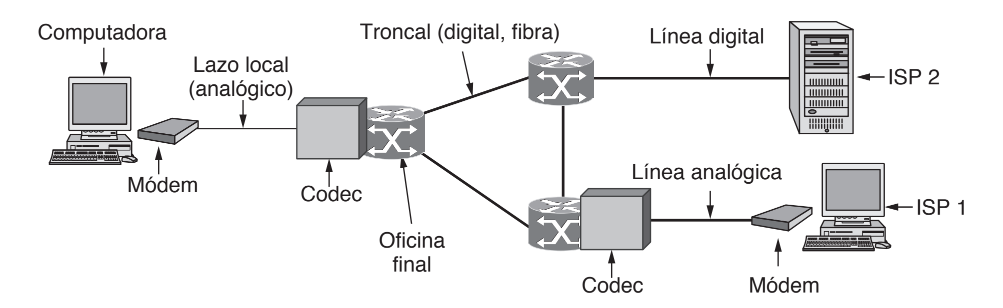


Figura 2-32. Empleo de transmisión tanto analógica como digital para una llamada de computadora a computadora. La conversión se realiza mediante los módems y los codecs.

Los módems telefónicos se utilizan para enviar bits entre dos computadoras a través de una línea telefónica de calidad de voz, en vez de la conversación que comúnmente llena la línea. La principal dificultad al hacer esto es que una línea telefónica de calidad de voz está limitada a 3100 Hz, justo lo suficiente para

transmitir una conversación. Este ancho de banda es más de cuatro veces menor al ancho de banda que se utiliza para Ethernet u 802.11 (WiFi). No es sorprendente que las tasas de transmisión de datos de los módems telefónicos también sean cuatro veces menores que las de Ethernet y 802.11.

Analicemos cifras para ver por qué ocurre esto. El teorema de Nyquist nos indica que incluso con una línea perfecta de 3 000 Hz (que en definitiva no puede ser una línea telefónica), no tiene caso enviar símbolos a una tasa mayor de 6 000 baudios. En la práctica, la mayoría de los módems envían a una tasa de 2400 símbolos/seg, o 2 400 baudios, y se enfocan en obtener múltiples bits por símbolo mientras permiten el tráfico en ambos sentidos y al mismo tiempo (mediante el uso de distintas frecuencias para direcciones diferentes).

El humilde módem de 2 400 bps utiliza 0 volts para un 0 lógico y 1 volt para un 1 lógico, con 1 bit por símbolo. Si lo mejoramos un poco, puede usar cuatro símbolos distintos, como en las cuatro fases de QSPK, de modo que con 2 bits/símbolo puede alcanzar una tasa de transmisión de datos de 4 800 bps.

A medida que la tecnología ha mejorado se ha logrado un gran progreso en las tasas de transmisión ahora son cada vez mayores. Las tasas más altas requieren un conjunto mayor de símbolos, o **constelación**. Al manejar muchos símbolos, incluso una pequeña cantidad de ruido en la amplitud o fase detectada puede producir un error. Para reducir la probabilidad de errores, los estándares para los módems de mayor velocidad utilizan algunos de los símbolos para la corrección de errores. A estos esquemas se les conoce como **TCM** (**Modulación Codificada de Enrejado**, del inglés *Tellis Coded Modulation*) (Ungerboeck, 1987).

El estándar de módem **V.32** utiliza 32 puntos de constelación para transmitir 4 bits de datos y 1 bit de verificación por símbolo a 2 400 baudios, para lograr 9 600 bps con corrección de errores. El siguiente nivel por encima de 9 600 bps es el de 14 400 bps. Se llama **V.32 bis** y transmite 6 bits de datos con 1 bit de verificación por símbolo, a 2 400 baudios. Después está el **V.34**, que logra 28 800 bps mediante la transmisión de 12 bits de datos/símbolo a 2 400 baudios. Ahora la constelación tiene miles de puntos. El último módem de esta serie es el **V.34 bis**, que utiliza 14 bits de datos/símbolo a 2 400 baudios para lograr 33 600 bps.

¿Por qué nos detenemos aquí? La razón por la que los módems estándar se detienen a 33 600 es que el límite de Shannon para el sistema telefónico es de alrededor de 35 kbps, con base en la longitud promedio de los lazos locales y la calidad de estas líneas. Una velocidad mayor a ésta violaría las leyes de la física (departamento de termodinámica).

Sin embargo, hay una manera de cambiar la situación. En la oficina final de la compañía telefónica, los datos se convierten a un formato digital para transmitirlos dentro de la red telefónica (el núcleo de la red telefónica los convirtió de análogos a digitales en etapas anteriores). El límite de 35 kbps es para la situación en la que hay dos lazos locales, uno en cada extremo. Cada uno de ellos agrega ruido a la señal. Si pudiéramos deshacernos de uno de estos lazos, podríamos aumentar la SNR y se duplicaría la tasa máxima de transmisión.

Esta propuesta es la encargada de hacer que funcionen los módems de 56 kbps. Uno de los extremos, por lo general un ISP, recibe una transmisión digital de alta calidad de la oficina final más cercana. Así, cuando un extremo de la conexión es una señal de alta calidad, como en la mayoría de los ISP actuales, la tasa máxima de transmisión de datos puede ser de hasta 70 kbps. Entre dos usuarios domésticos con módems y líneas analógicas, el máximo sigue siendo de 33.6 kbps.

La razón por la que se utilizan módems de 56 kbps (en vez de 70 kbps) está relacionada con el teorema de Nyquist. Para transportar un canal telefónico dentro del sistema telefónico se usan muestras digitales. Cada canal telefónico tiene 4 000 Hz de amplitud cuando se incluyen las bandas de guarda. Por ende, el número de muestras por segundo necesario para reconstruirlo es de 8 000. El número de bits por muestra en Estados Unidos es de 8, de los cuales uno se puede usar para propósitos de control, con lo cual obtenemos 56 000 bits/seg de datos para el usuario. En Europa los 8 bits están disponibles para los

usuarios, por lo que se podrían haber utilizado módems de 64 000 bits/seg, pero para acordar un estándar internacional se eligió el valor de 56 000.

El resultado final son los estándares de módems **V.90** y **V.92**, que proveen un canal descendente de 56 kbps (del ISP al usuario) y canales ascendentes de 33.6 kbps y 48 kbps (del usuario al ISP), respectivamente. La asimetría se debe a que por lo general se transportan más datos del ISP al usuario que de la otra forma. También significa que se puede asignar más del ancho de banda limitado al canal descendente para incrementar la probabilidad de que pueda funcionar a 56 kbps.

Líneas de suscriptor digital

Cuando por fin la industria telefónica logró los 56 kbps, se dio unas palmadas en la espalda por un trabajo bien hecho. Mientras tanto, la industria de la TV por cable ofrecía velocidades de hasta 10 Mbps en cables compartidos. A medida que el acceso a Internet se convirtió en una parte cada vez más importante de su negocio, las compañías telefónicas (LEC) se dieron cuenta de que necesitaban un producto más competitivo. La respuesta fue ofrecer nuevos servicios digitales a través del lazo local.

Al principio hubo muchos ofrecimientos de alta velocidad que se traslapaban, todos bajo el nombre general de **xDSL** (**Línea de Suscriptor Digital**, del inglés *Digital Subscriber Line*), por las diversas *x*. Los servicios con mayor ancho de banda que el servicio telefónico estándar generalmente se denominan **banda ancha**, aunque en realidad el término es más un concepto de marketing que un concepto técnico específico. Más adelante hablaremos sobre el que se ha convertido en el más popular de estos servicios, **ADSL** (**DSL asimétrico**, del inglés *Asymetric DSL*). También utilizaremos el término DSL o xDSL como abreviatura para todas las variedades de esta tecnología.

La razón por la que los módems son tan lentos es que los teléfonos se inventaron para transportar la voz humana, por lo que todo el sistema se optimizó cuidadosamente para este fin. Los datos siempre han sido como “hijastros”. En el punto en el que cada lazo local termina en la oficina final, el cable pasa por un filtro que atenúa todas las frecuencias menores de 300 Hz y mayores de 3 400 Hz. El corte no es muy abrupto (300 Hz y 3 400 Hz son los puntos de 3 dB), por lo que el ancho de banda se indica generalmente como 4 000 Hz, aun cuando la distancia entre los puntos de 3 dB sea de 3100 Hz. Por lo tanto, los datos en el cable también están restringidos a esta banda estrecha.

El truco para que xDSL funcione es que cuando un cliente se suscribe al servicio, la línea entrante se conecta a un tipo distinto de conmutador que no tiene este filtro y, en consecuencia, toda la capacidad del lazo local queda disponible. En este caso, el factor limitante se convierte en el medio físico del lazo local, que soporta aproximadamente 1 MHz, no el ancho de banda artificial de 3100 Hz creado por el filtro.

Por desgracia, la capacidad del lazo local disminuye rápidamente con la distancia desde la oficina final, puesto que la señal se degrada cada vez más a lo largo del cable. Esto también depende del grosor y la calidad general del par trenzado. En la figura 2-33 se muestra un gráfico del ancho de banda potencial en función de la distancia. Esta figura supone que todos los demás factores son óptimos (nuevos cables, haces modestos, etcétera).

La implicación de esta figura crea un problema para la compañía telefónica. Cuando selecciona la velocidad a ofrecer, elije al mismo tiempo un radio desde sus oficinas finales, más allá del cual no se podrá ofrecer el servicio. Esto significa que cuando los clientes distantes traten de contratar el servicio, es posible que se les diga: “Gracias por su interés, pero usted vive a una distancia 100 metros más allá de la oficina final más cercana para obtener este servicio. ¿Se podría mudar?”. Entre menor velocidad se seleccione, mayor será el radio y se podrán cubrir más clientes. Pero entre menor sea la velocidad, menos atractivo será el servicio y menos personas estarán dispuestas a pagar por él. Aquí es donde los negocios se encuentran con la tecnología.

Todos los servicios xDSL se diseñaron con ciertos objetivos en mente. Primero, los servicios deben trabajar sobre los lazos locales existentes de par trenzado categoría 3. Segundo, no deben afectar a los

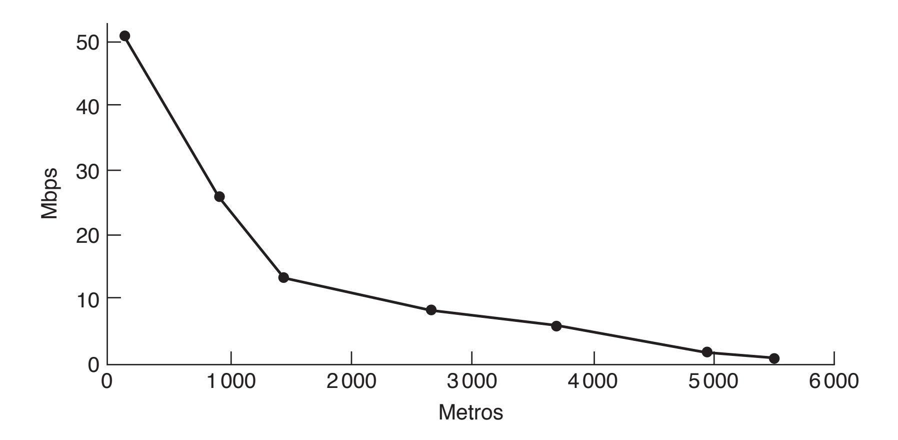


Figura 2-33. Ancho de banda *vs*. distancia sobre cable UTP categoría 3 para DSL.

teléfonos o máquinas de fax existentes de los clientes. Tercero, deben ser mucho más veloces que 56 kbps. Cuarto, siempre deben estar encendidos y cobrar una cuota mensual, sin cargos por minuto.

Para cumplir con los objetivos técnicos, el espectro disponible de 1.1 MHz en el lazo local se divide en 256 canales independientes de 4312.5 Hz cada uno. Este arreglo se muestra en la figura 2-34. El esquema OFDM, que vimos en la sección anterior, se utiliza para enviar datos a través de estos canales, aunque a menudo se le conoce como **DMT** (**Multitono Discreto**, del inglés *Discrete MultiTone*) en el contexto de ADSL. El canal 0 se utiliza para **POTS** (**Servicio Telefónico Tradicional**, del inglés *Plain Old Telephone Service*). Los canales 1 al 5 no se utilizan para evitar que las señales de voz y datos interfieran entre sí. De los 250 canales restantes, uno se utiliza para el control ascendente y otro para el control descendente. El resto están disponibles para los datos del usuario.

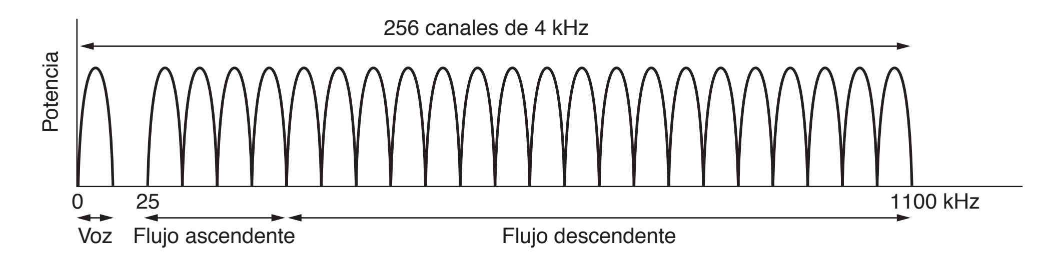


Figura 2-34. Operación de ADSL usando modulación por multitono discreta.

En principio, cada uno de los canales restantes se puede utilizar para un flujo de datos *full-dúplex*, pero las armónicas, la diafonía y otros efectos mantienen a los sistemas prácticos muy por debajo del límite teórico. El proveedor es el encargado de determinar cuántos canales se van a utilizar para el flujo ascendente y cuántos para el flujo descendente. Es técnicamente posible usar una mezcla 50/50 de flujo ascendente y flujo descendente, pero la mayoría de los proveedores asignan algo así como 80% a 90% del ancho de banda para el canal de flujo descendente, ya que casi todos los usuarios descargan más datos de los que envían. Esta elección es la razón de la "A" en ADSL. Una división común es 32 canales para el flujo ascendente y el resto para el flujo descendente. También es posible que unos cuantos de los canales de flujo ascendente más altos sean bidireccionales para incrementar el ancho de banda, aunque para hacer esta optimización se requiere agregar un circuito especial para cancelar ecos.

En 1999 se aprobó el estándar internacional ADSL, conocido como **G.dmt**. Este estándar permite velocidades de hasta 8 Mbps para el flujo descendente y 1 Mbps para el flujo ascendente. En 2002 se sustituyó por una segunda generación, conocida como ADSL2, con varias mejoras para permitir velocidades de hasta 12 Mbps para el flujo descendente y 1 Mbps para el flujo ascendente. Ahora tenemos el ADSL2+ que duplica la velocidad para el flujo descendente hasta 24 Mbps, para lo cual duplica el ancho de banda y utiliza 2.2 Mhz a través del par trenzado.

No obstante, los números aquí citados son las velocidades en el mejor de los casos para líneas buenas que estén cerca (en un rango de 1 a 2 km) de la central. Pocas líneas soportan estas tasas de transmisión y pocos proveedores ofrecen estas velocidades. Por lo general, los proveedores ofrecen algo así como 1 Mbps para el flujo descendente y 256 kbps para el flujo ascendente (servicio estándar), 4 Mbps para el flujo descendente y 1 Mbps para el flujo ascendente (servicio mejorado), y 8 Mbps para el flujo descendente con 2 Mbps para el flujo ascendente (servicio de primera).

Dentro de cada canal se utiliza la modulación QAM a una tasa aproximada de 4000 símbolos/seg. La calidad de la línea en cada canal se monitorea de manera constante y la tasa de transmisión de datos se ajusta mediante el uso de una constelación más grande o más pequeña, como las de la figura 2-23. Los distintos canales pueden tener diferentes tasas de transmisión de datos, pudiendo enviar hasta 15 bits por símbolo en un canal con una SNR alta, y descender hasta 2,1 o ningún bit por símbolo en un canal con una SNR baja, dependiendo del estándar.

En la figura 2-35 se muestra un arreglo ADSL común. En este esquema, el técnico de una compañía telefónica debe instalar un **NID** (**Dispositivo de Interfaz de Red**, del inglés *Network Interface Device*) en la residencia del cliente. Esta pequeña caja de plástico marca el fin de la propiedad de la compañía telefónica y el inicio de la propiedad del cliente. Cerca del NID (o algunas veces combinado con él) hay un **divisor** o **splitter**, un filtro analógico que separa la banda de 0 a 4000 Hz que utiliza el POTS de los datos. La señal del POTS se encamina al teléfono o máquina de fax existente. La señal de datos se encamina a un módem ADSL, el cual utiliza el procesamiento digital de señales para implementar el OFDM. Como la mayoría de los módems ADSL son externos, es necesario conectarlos a la computadora mediante una conexión de alta velocidad. Por lo común esto se hace mediante Ethernet, un cable USB o una red 802.11.

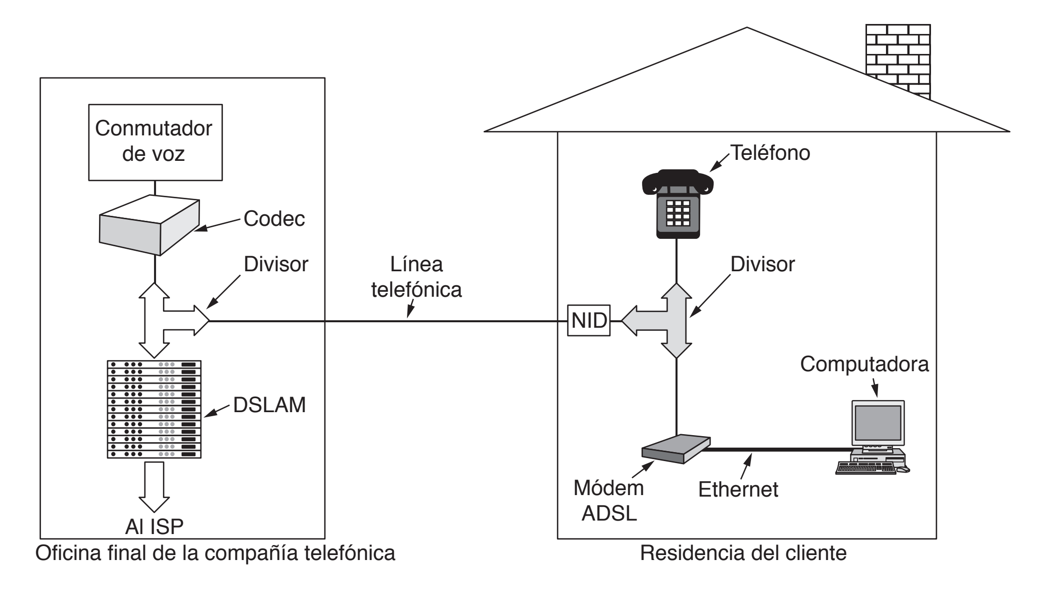


Figura 2-35. Configuración típica de un equipo ADSL.

Al otro extremo del cable, del lado de la oficina final, se instala un divisor correspondiente. Aquí se filtra y elimina la porción de la señal correspondiente a la voz, y se envía al conmutador de voz normal. La señal por encima de 26 kHz se encamina a un nuevo dispositivo conocido como **DSLAM** (**Multiplexor de Acceso a la Línea de Suscriptor Digital**, del inglés *Digital Subscriber Line Access Multiplexer*), el cual contiene el mismo tipo de procesador digital de señales que el módem ADSL. Una vez que se recuperan los bits de la señal, se forman paquetes y se envían al ISP.

Esta completa separación entre el sistema de voz y ADSL facilita de manera relativa el hecho de que una compañía telefónica pueda implementar esta tecnología. Todo lo que se necesita es comprar un DSLAM y un divisor, y conectar a los suscriptores de ADSL al divisor. Otros servicios de alto ancho de banda (como ISDN) requieren cambios mucho más grandes en el equipo de conmutación existente.

Una desventaja del diseño de la figura 2-35 es la necesidad de un NID y un divisor en la residencia del cliente. Sólo un técnico de la compañía telefónica puede instalarlos, de modo que hay que enviar un técnico a la residencia del cliente y esto genera un costo considerable. En respuesta a este problema se estandarizó también un diseño alternativo sin divisor, conocido como **G.lite**. Es igual que la figura 2-35, sólo que sin el divisor del cliente. La línea telefónica existente se utiliza así como está. La única diferencia es que se debe insertar un microfiltro en cada terminal telefónica, entre el teléfono o módem ADSL y el cable. El microfiltro para el teléfono es un filtro pasabajas que elimina las frecuencias superiores a 3 400 Hz; el microfiltro para el módem ADSL es un filtro pasaaltas que elimina las frecuencias inferiores a 26 kHz. Sin embargo, este sistema no es tan confiable como el de divisor, por lo que sólo se puede usar el G.lite a una velocidad de hasta 1.5 Mbps (en comparación con los 8 Mbps de la ADSL con un divisor). Para obtener más información sobre ADSL, consulte a Starr (2003).

Fibra para el hogar

Los lazos locales de cobre existentes limitan el desempeño de ADSL y los módems telefónicos. Para proveer servicios de red más rápidos y mejores, las compañías telefónicas están actualizando los lazos locales en cada oportunidad mediante la instalación de fibra óptica en todo el camino hasta las casas y oficinas. Al resultado se le conoce como **FTTH** (**Fibra para el Hogar**, del inglés *Fiber To The Home*). Aunque esta tecnología ha estado ya disponible durante cierto tiempo, la instalación en sí apenas empezó a despegar en 2005 con el aumento en la demanda de Internet de alta velocidad por parte de los clientes que usaban DSL y cable, que querían descargar películas. Cerca de 4% de los hogares en Estados Unidos están ahora conectados a FTTH con velocidades de acceso a Internet de hasta 100 Mbps.

Existen algunas variaciones de la forma “FTTX” (en donde X puede ser sótano, banqueta o vecindario), las cuales se utilizan para indicar que la fibra óptica puede estar instalada cerca del hogar. En este caso, el cobre (par trenzado o cable coaxial) provee velocidades suficientemente rápidas durante el último tramo corto. La elección de hasta dónde tender la fibra es meramente económica, ya que se balancea el costo con los ingresos esperados. En cualquier caso, el punto es que la fibra óptica ha cruzado la barrera tradicional de la “última milla”. En nuestro análisis nos enfocaremos en la FTTH.

Al igual que su antecesor, el lazo local de cables de cobre, el lazo local de fibra óptica es pasivo. Esto significa que no se requiere equipo energizado para amplificar o procesar las señales de alguna otra forma. La fibra simplemente transporta las señales entre el hogar y la oficina final. Esto a su vez reduce el costo y mejora la confiabilidad.

Por lo general, las fibras que salen de los hogares se unen de manera que llegue una sola fibra a la oficina final por cada grupo de hasta 100 casas. En el sentido del flujo descendente hay divisores ópticos que dividen la señal que proviene de la oficina final, de modo que llegue a todas las casas. Se requiere un cifrado para la seguridad si sólo una casa puede decodificar la señal. En sentido del flujo ascendente hay combinadores ópticos que mezclan las señales que salen de las casas en una sola señal que se recibe en la oficina final.

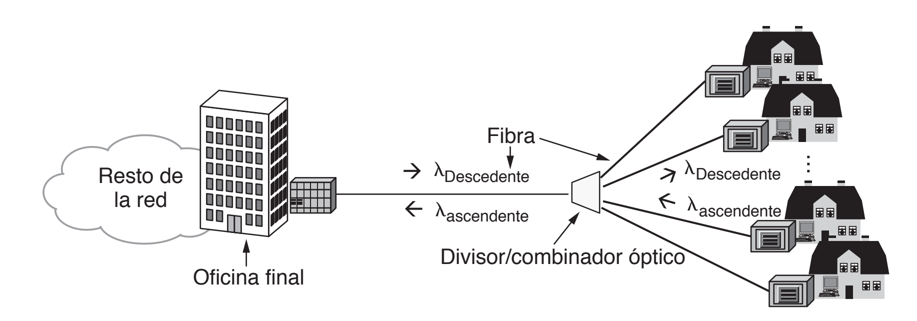


Figura 2-36. Red óptica pasiva utilizada en la fibra hasta el hogar.

Esta arquitectura se llama **PON** (**Red Óptica Pasiva**, del inglés *Passive Optical Network*) y se muestra en la figura 2-36. Es común utilizar una longitud de onda compartida entre todas las casas para la transmisión del flujo descendente y otra longitud de onda para la transmisión del flujo ascendente.

Incluso con la división, gracias al tremendo ancho de banda y a la baja atenuación de la fibra óptica, las PON pueden ofrecer altas tasas de transmisión a los usuarios a través de distancias de hasta 20 km. Las tasas de transmisión de datos reales y otros detalles dependen del tipo de PON. Hay dos tipos comunes. Las **GPON** (**PON con capacidad de Gigabit**, del inglés *Gigabit-capable PONs*) vienen del mundo de las telecomunicaciones, por lo que se definen mediante un estándar de la ITU. Las **EPON** (**PON Ethernet**) están más sintonizadas con el mundo de las redes, por lo que se definen mediante un estándar del IEEE. Ambas operan a una velocidad cercana a un gigabit y pueden transportar tráfico para distintos servicios, incluyendo Internet, video y voz. Por ejemplo, las GPON proveen un flujo descendente de 2.4 Gbps y un flujo ascendente de 1.2 o 2.4 Gbps.

Se necesita un protocolo para compartir la capacidad de la fibra individual en la oficina final, entre las distintas casas. El sentido del flujo descendente es fácil. La oficina final puede enviar mensajes a cada una de las distintas casas en el orden que desee. Pero en sentido del flujo ascendente no se pueden enviar mensajes de distintas casas al mismo tiempo pues habría colisiones. Las casas tampoco pueden escuchar las transmisiones de las demás casas, por lo que no pueden escuchar antes de transmitir. La solución es que el equipo en las casas solicite y reciba ranuras de tiempo para usar la línea al equipo en la oficina final. Para que esto funcione, hay un proceso de variación para ajustar los tiempos de transmisión de las casas, de modo que todas las señales que se reciban en el extremo de la oficina final estén sincronizadas. El diseño es similar al de los módems de cable, que veremos más adelante en este capítulo. Para obtener más información sobre el futuro de las PON, consulte a Grobe y Elbers (2008).

2.6.4 Troncales y multiplexión

Las troncales en la red telefónica no sólo son mucho más rápidas que los lazos locales, sino que son diferentes en otros dos aspectos. El núcleo de la red telefónica transporta información digital, no información analógica; es decir, bits y no voz. Para esto se necesita una conversión en la oficina final a un formato digital para transmitir la señal a través de las troncales de largo recorrido. Estas troncales transportan miles, incluso millones, de llamadas al mismo tiempo. Esta compartición es muy importante para obtener economía de escala, ya que en esencia cuesta lo mismo instalar y mantener una troncal con alto ancho de banda que una con bajo ancho de banda entre dos oficinas de conmutación. Esto se logra mediante versiones de multiplexión TDM y FDM.

A continuación analizaremos brevemente cómo se digitalizan las señales de voz, de modo que se puedan transportar mediante la red telefónica. Después veremos cómo se utiliza la TDM para transportar

bits en troncales, incluyendo el sistema TDM que se utiliza para las fibras ópticas (SONET). Después pasaremos a la FDM y su aplicación en la fibra óptica, o multiplexión por división de longitud de onda.

Digitalización de las señales de voz

En las primeras etapas del desarrollo de la red telefónica, el núcleo manejaba las llamadas de voz como información analógica. Las técnicas de FDM se utilizaron durante muchos años para multiplexar los canales de voz de 4 000 Hz (compuestos de 3100 Hz más las bandas de guarda) en unidades cada vez más grandes. Por ejemplo, un **grupo** está formado de 12 llamadas en la banda de 60 kHz a 108 kHz, cinco grupos (un total de 60 llamadas) constituyen un **supergrupo**, y así en lo sucesivo. Estos métodos de FDM se siguen utilizando en algunos cables de cobre y canales de microondas. Sin embargo, la FDM requiere circuitos analógicos y no es posible que una computadora realice esta tarea. En contraste, la TDM se puede manejar en su totalidad mediante componentes electrónicos digitales, por lo que se ha vuelto muy popular en años recientes. Como la TDM sólo se puede utilizar para datos digitales y los lazos locales producen señales analógicas, se necesita un convertidor de analógico a digital en la oficina final, en donde todos los lazos locales se unen para combinarlos en troncales salientes.

Las señales analógicas se digitalizan en la oficina final mediante un dispositivo llamado **codec** (abreviación de "*co*dificador-*dec*odificador"), el cual realiza 8 000 muestreos por segundo (125 μseg/muestra), ya que el teorema de Nyquist establece que son suficientes para capturar toda la información del ancho de banda de un canal telefónico de 4 kHz. A una velocidad de muestreo menor se perdería información; a una más alta no se obtendría información adicional. Cada muestra de la amplitud de la señal se cuantifica en un número de 8 bits.

A esta técnica se le conoce como **PCM** (**Modulación de Codificación de Impulsos**, del inglés *Pulse Code Modulation*) y constituye el corazón del sistema telefónico moderno. Como consecuencia, virtualmente todos los intervalos dentro del sistema telefónico son múltiplos de 125 μseg. Así, la tasa de transmisión estándar de datos sin comprimir para una llamada telefónica de voz es de 8 bits cada 125 μseg, o de 64 kbps.

En el otro extremo de la llamada se recrea una señal analógica a partir de las muestras cuantificadas, para lo cual éstas se reproducen (y suavizan) en el tiempo. La señal que se obtenga no será exactamente igual a la señal analógica original aun cuando hayamos muestreado según la tasa de Nyquist, ya que las muestras se cuantificaron. Para reducir el error debido a la cuantificación, los niveles de cuantificación se separan en forma irregular. Se utiliza una escala logarítmica que produce relativamente más bits para amplitudes de señal más pequeñas, y relativamente menos bits para amplitudes de señal más grandes. De esta forma, el error es proporcional a la amplitud de la señal.

Hay dos versiones de cuantificación muy populares: la **ley** μ (μ-*law*) que se utiliza en Estados Unidos y Japón, y la **ley A** (*A-law*), que se utiliza en Europa y el resto del mundo. Ambas versiones se especifican en el estándar ITU G.711. Una manera equivalente de pensar sobre este proceso es imaginar que el rango dinámico de la señal (o la proporción entre el mayor y el menor valor posible) se comprime antes de ser cuantificado (de manera uniforme), y que después se expande al recrear la señal analógica. Por esta razón se le conoce como **compansión** (*companding*). También es posible comprimir las muestras después de digitalizarlas, de modo que se requieran mucho menos de 64 kbps. Sin embargo, dejaremos este tema para cuando exploremos las aplicaciones de audio tales como voz sobre IP.

Multiplexión por división de tiempo

La TDM basada en PCM se utiliza para transmitir varias llamadas de voz a través de troncales, para lo cual se envía una muestra de cada llamada por cada 125 μseg. Cuando la transmisión digital empezó a emerger como una tecnología viable, la ITU (que en ese entonces se llamaba CCITT) no pudo llegar a un

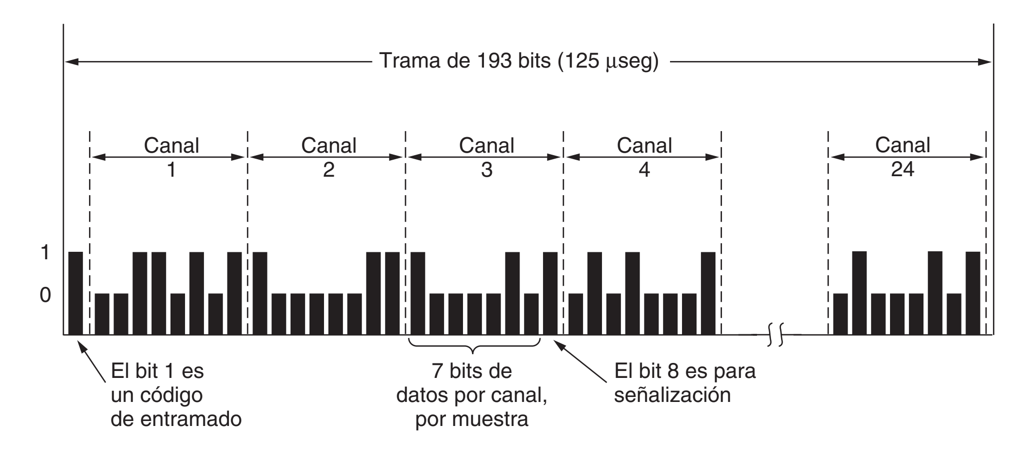


Figura 2-37. La portadora T1 (1.544 Mbps).

acuerdo sobre un estándar internacional para PCM. En consecuencia, ahora hay una variedad de esquemas incompatibles que se utilizan en distintos países alrededor del mundo.

El método que se utiliza en Norteamérica y Japón es la portadora **T1**, que se muestra en la figura 2-37 (hablando en sentido técnico, el formato se llama DS1 y la portadora se llama T1, pero para seguir la popular tradición industrial no haremos esta ligera distinción aquí). La portadora T1 consiste en 24 canales de voz multiplexados entre sí. A su vez, cada uno de los 24 canales puede insertar 8 bits en el flujo de salida.

Una trama consiste en $24 \times 8 = 192$ bits más un bit extra para fines de control, con lo cual se obtienen 193 bits cada 125 μseg. Esto nos da una tasa de transmisión de datos bruta aproximada de 1.544 Mbps, de los cuales se utilizan 8 kbps para señalización. El bit 193 se utiliza para la sincronización de la trama y señalización. En una variación, el bit 193 se utiliza a través de un grupo de 24 tramas llamado **supertrama extendida**. Seis de los bits, en las posiciones número 4, 8, 12, 16, 20 y 24, siguen el patrón alternativo 001011…. Por lo general, el receptor verifica en forma continua este patrón para asegurarse de que no haya perdido la sincronización. Se utilizan seis bits más para enviar un código de verificación de errores, lo cual ayuda a que el receptor confirme que está sincronizado. Si se sale de sincronización, el receptor puede explorar el patrón y validar el código de verificación de errores para volverse a sincronizar. Los 12 bits restantes se utilizan para la información de control necesaria para mantener y operar la red, como el informe del desempeño del extremo remoto.

El formato T1 tiene algunas variaciones. Las primeras versiones enviaban la información de señalización **dentro de banda** (**in-band**), lo cual significa que está en el mismo canal que los datos y, por ende, se usan algunos de los bits de datos. Este diseño es una forma de **señalización asociada al canal**, ya que cada canal tiene su propio subcanal privado de señalización. En un arreglo, el bit menos significativo de una muestra de 8 bits en cada canal se utiliza en cada sexta trama. Este arreglo tiene el pintoresco nombre de **señalización por robo de bit**. La idea es que unos cuantos bits robados no importan en las llamadas de voz. Nadie escuchará la diferencia.

Pero para los datos es otra historia. Sería inútil entregar los bits incorrectos, por no decir otra cosa peor. Si se utilizan versiones antiguas de T1 para transportar datos, sólo se pueden usar 7 de 8 bits, o 56 kbps, en cada uno de los 24 canales. Por el contrario, las versiones más recientes de T1 proveen canales claros en los que se pueden utilizar todos los bits para enviar datos. Canales limpios es lo que desean los negocios que rentan una línea T1 cuando se trata de enviar datos a través de la red telefónica, en vez de

muestras de voz. Así, la señalización para cualquier llamada de voz se maneja **fuera de banda**, o en un canal separado de los datos. Por lo regular la señalización se lleva a cabo mediante la **señalización de canal común**, en donde hay un canal de señalización compartido. Para este fin se puede utilizar uno de los 24 canales.

Fuera de Norteamérica y de Japón se utiliza la portadora **E1** de 2.048 Mbps en vez de T1. Esta portadora tiene 32 muestras de datos de 8 bits empaquetadas en la trama básica de 125 μseg. Treinta canales se utilizan para información y se pueden usar hasta dos para señalización. Cada grupo de cuatro tramas proporciona 64 bits de señalización, de los cuales la mitad se utilizan para señalización (ya sea asociada al canal o de canal común) y la otra mitad se utilizan para sincronización de la trama o se reservan para que cada país los utilice según sus necesidades.

La multiplexión por división de tiempo nos permite multiplexar varias portadoras T1 en portadoras de orden más alto. La figura 2-38 muestra cómo se puede hacer esto. A la izquierda vemos cuatro canales T1 que se multiplexan en un canal T2. La multiplexión en T2 y las portadoras superiores se realiza bit por bit, en vez de byte por byte como con los 24 canales de voz que forman una trama T1. Cuatro flujos T1 a 1.544 Mbps deberían generar 6.176 Mbps, pero en realidad T2 es de 6.312 Mbps. Los bits adicionales se utilizan para entramar y para recuperación en caso de que la portadora pierda sincronía. Los clientes utilizan mucho T1 y T3, mientras que T2 y T4 sólo se utilizan dentro del mismo sistema telefónico, de modo que no son muy conocidos.

En el siguiente nivel, siete flujos T2 se combinan a nivel de bits para formar un flujo T3. Después se unen seis flujos T3 para formar un flujo T4. En cada paso se agrega una pequeña cantidad de sobrecarga para el entramado y la recuperación, en caso de que se pierda la sincronización entre el emisor y el receptor.

Así como hay un desacuerdo en cuanto a la portadora básica entre Estados Unidos y el resto del mundo, también hay un desacuerdo en cuanto a cómo se debe multiplexar en portadoras con mayor ancho de banda. El esquema de Estados Unidos de avanzar en pasos de 4, 7 y 6 no pareció lógico a todo el mundo, de modo que el estándar de la ITU prescribe la multiplexión de cuatro flujos en un flujo en cada nivel. Además, los datos de entramado y de recuperación son distintos en los estándares de Estados Unidos y de la ITU. La jerarquía de la ITU para 32, 128, 512, 2048 y 8192 canales funciona a velocidades de 2.048, 8.848, 34.304, 139.264 y 565.148 Mbps.

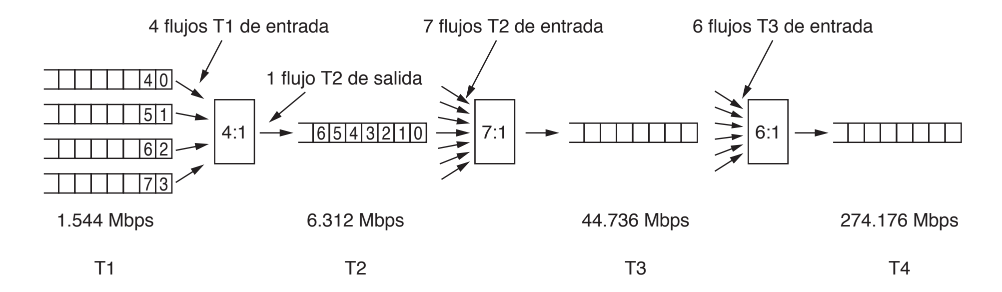


Figura 2-38. Multiplexión de flujos T1 en portadoras más altas.

SONET/SDH

En los primeros días de la fibra óptica, cada compañía telefónica tenía su propio sistema TDM óptico patentado. Después de la división de AT&T en 1984, las compañías telefónicas locales se tuvieron que conectar a múltiples portadoras de larga distancia, todas con diferentes sistemas TDM ópticos, por lo

que se hizo obvia la necesidad de la estandarización. En 1985, Bellcore (la rama de investigación de las RBOC) empezó a trabajar en un estándar llamado **SONET** (**Red Óptica Síncrona**, del inglés *Synchronous Optical NETwork*).

Más adelante la ITU se unió al esfuerzo, que en 1989 produjo un estándar llamado SONET y un conjunto de recomendaciones paralelas de la ITU (G.707, G.708 y G.709). Las recomendaciones de la ITU se llaman **SDH** (**Jerarquía Digital Síncrona**, del inglés *Synchonous Digital Hierarchy*) pero difieren de SONET sólo en pequeños aspectos. Virtualmente todo el tráfico telefónico de larga distancia en Estados Unidos (además de una gran parte del mismo en otros países) utiliza ahora troncales que funcionan con SONET en la capa física. Para obtener información adicional sobre SONET, consulte a Bellamy (2000), Goralski (2002) y Shepard (2001).

El diseño de SONET tenía cuatro objetivos principales. Primero que nada, SONET tenía que hacer posible la interconexión de redes de distintas portadoras. Para lograr esta meta se requería definir un estándar de señalización común respecto a la longitud de onda, la sincronización, la estructura de tramas y otras cuestiones.

En segundo lugar, se necesitaba una manera de unificar los sistemas digitales estadounidense, europeo y japonés, los cuales se basaban en canales PCM de 64 kbps pero los combinaban de formas distintas (e incompatibles).

En tercer lugar, SONET tenía que proporcionar una manera de multiplexar varios canales digitales. En el momento en que se creó SONET, la portadora digital de mayor velocidad utilizada ampliamente en Estados Unidos era T3, a 44.736 Mbps. La T4 se había definido pero no se utilizaba mucho, y no había nada definido más allá de la velocidad de la T4. Una parte de la misión de SONET era continuar la jerarquía hasta los gigabits/seg y más allá. También se necesitaba una manera estándar de multiplexar canales más lentos en un solo canal SONET.

En cuarto lugar, SONET tenía que proporcionar soporte para las operaciones, la administración y el mantenimiento (OAM) necesarios para administrar la red. Los sistemas anteriores no hacían esto muy bien.

Una decisión temprana fue la de convertir a SONET en un sistema TDM tradicional, con todo el ancho de banda de la fibra óptica dedicado a un canal que contuviera ranuras de tiempo para los distintos subcanales. Como tal, SONET es un sistema síncrono. Cada emisor y receptor está enlazado a un reloj común. El reloj maestro que controla al sistema tiene una precisión de alrededor de 1 en 10^9. Los bits en una línea SONET se envían a intervalos extremadamente precisos, controlados por el reloj maestro.

La trama SONET básica es un bloque de 810 bytes que se emite cada 125 μseg. Puesto que SONET es síncrona, las tramas se emiten haya o no datos útiles que enviar. La velocidad de 8 000 tramas/seg coincide exactamente con la tasa de muestreo de los canales PCM que se utilizan en todos los sistemas de telefonía digital.

Las tramas SONET de 810 bytes se pueden describir mejor como un rectángulo de bytes de 90 columnas de ancho por 9 filas de alto. De esta forma, $8 \times 810 = 6480$ bits se transmiten 8 000 veces por segundo, lo cual nos da una tasa de datos aproximada de 51.84 Mbps. Esta distribución es el canal SONET básico, llamado **STS-1** (**Señal Síncrona de Transporte 1**, del inglés *Synchonous Transport Signal-1*). Todas las troncales SONET son múltiplos de STS-1.

Las primeras tres columnas de cada trama se reservan para información administrativa del sistema, como se ilustra en la figura 2-39. En este bloque las primeras tres filas contienen el encabezado de la sección (*section overhead*); las siguientes seis contienen el encabezado de la línea (line overhead). El encabezado de la sección se genera y se comprueba al inicio y al final de cada sección, mientras que el encabezado de la línea se genera y comprueba al inicio y al final de cada línea.

Un transmisor SONET envía tramas de 810 bytes consecutivos, sin espacios vacíos entre ellas, aun cuando no haya datos (en cuyo caso envía datos ficticios). Desde el punto de vista del receptor todo lo que ve es un flujo continuo de bits, de modo que, ¿cómo sabe en dónde empieza cada trama? La respuesta

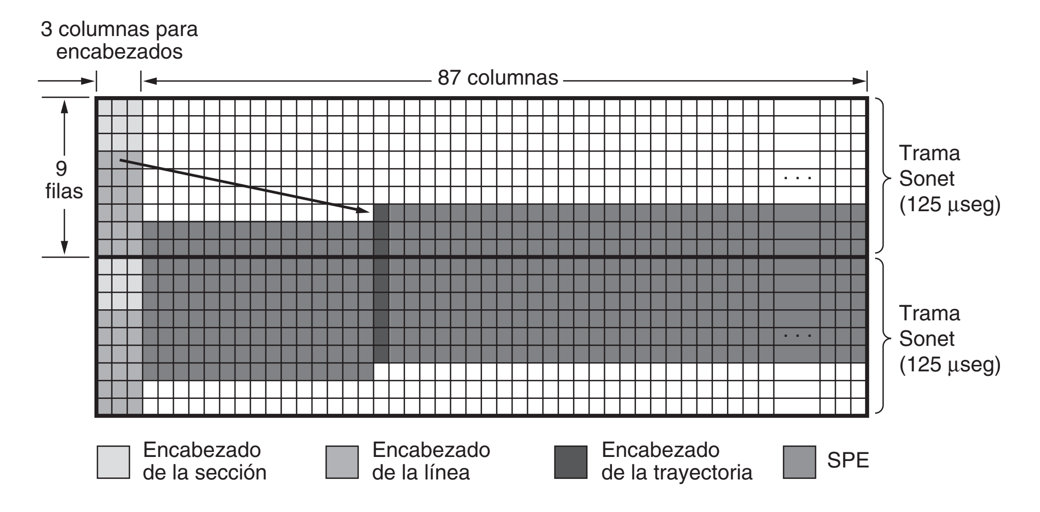


Figura 2-39. Dos tramas SONET consecutivas.

es que los primeros 2 bytes de cada trama contienen un patrón fijo que el receptor busca. Si encuentra este patrón en el mismo lugar en una gran cantidad de tramas consecutivas, asume que está sincronizado con el emisor. En teoría, un usuario podría insertar este patrón en la carga útil de una manera regular, pero en la práctica no es posible debido al multiplexado de múltiples usuarios en la misma trama, además de otras razones.

Las 87 columnas restantes de cada trama contienen $87 \times 9 \times 8 \times 8000 = 50.112$ Mbps de datos de usuario, que podrían ser muestras de voz, T1 y otras portadoras completas, o paquete. SONET es simplemente un contenedor conveniente para transportar bits. La **SPE** (**Contenedor de Carga Útil Síncrona**, del inglés *Synchronous Pyload Envelope*), que transporta los datos de usuario, no siempre empieza en la fila 1, columna 4. Puede empezar en cualquier parte dentro de la trama. Hay un apuntador al primer byte de la primera fila del encabezado de la línea. La primera columna de la SPE es el encabezado de la trayectoria (es decir, el encabezado para el protocolo de subcapa de la trayectoria de extremo a extremo).

La facultad de que la SPE empiece en cualquier parte dentro de la trama SONET y de abarcar incluso dos tramas, como se muestra en la figura 2-39, otorga una flexibilidad adicional al sistema. Por ejemplo, si una carga útil llega al origen mientras que se está construyendo una trama SONET ficticia, se puede insertar en la trama actual en vez de retenerla hasta el inicio de la siguiente trama.

En la figura 2-40 se muestra la jerarquía de multiplexión de SONET/SDH. Se definieron tasas de STS-1 a STS-768, que varían cerca de una línea T3 a 40 Gbps. Es muy probable que se definan tasas aún mayores con el transcurso del tiempo, en donde la OC-3072 a 160 Gbps es la próxima en la línea, sólo si llega a ser tecnológicamente viable. La portadora óptica que corresponde a la STS-*n* se llama OC-*n* y es idéntica bit por bit, excepto que se necesita cierto reordenamiento de bits para la sincronización. Los nombres de SDH son diferentes y empiezan en OC-3 debido a que los sistemas basados en la ITU no tienen una tasa de transmisión cercana a los 51.84 Mbps. En la figura mostramos las tasas de transmisión comunes, que progresan a partir de OC-3 en múltiplos de cuatro. La tasa de transmisión de datos bruta incluye toda la cabecera. La tasa de transmisión de datos de SPE excluye los encabezados de línea y de sección. La tasa de transmisión de datos de usuario excluye todos los encabezados y sólo cuenta las 87 columnas para la carga útil.

Por cierto, cuando no se multiplexa una portadora como la OC-3, sino que transporta los datos de una sola fuente, se agrega la letra *c* (de concatenado) a la designación, de modo que OC-3c indica un flujo de datos proveniente de una fuente única a 155.52 Mbps. Los tres flujos OC-1 dentro de un flujo OC-3c se

SONET		SDH	Tasa de datos (Mbps)		
Eléctrica	Óptica	Óptica	Bruta	SPE	De usuario
STS-1	OC-1		51.84	50.112	49.536
STS-3	OC-3	STM-1	155.52	150.336	148.608
STS-12	OC-12	STM-4	622.08	601.344	594.432
STS-48	OC-48	STM-16	2488.32	2405.376	2377.728
STS-192	OC-192	STM-64	9953.28	9621.504	9510.912
STS-768	OC-768	STM-256	39813.12	38486.016	38043.648

Figura 2-40. Tasas de multiplexión de SONET y SDH.

entrelazan por columnas (primero la columna 1 del flujo 1, después la columna 1 del flujo 2, después la columna 1 del flujo 3, seguidas de la columna 2 del flujo 1, etc.) para obtener una trama de 270 columnas de anchura y 9 filas de profundidad.

Multiplexión por división de longitud de onda

Existe una forma de multiplexión por división de frecuencia es utilizada así como TDM para aprovechar el enorme ancho de banda de los canales de fibra óptica, la cual se llama **WDM** (**Multiplexión por División de Longitud de Onda**, del inglés *Wave-length Division Multiplexing*). El principio básico de la WDM en fibra óptica se ilustra en la figura 2-41. En este ejemplo se juntan cuatro fibras en un combinador óptico, cada una con su energía presente en una longitud de onda distinta. Los cuatro haces se combinan en una sola fibra compartida para transmitirla a un destino distante. En el extremo lejano, el haz se divide en la cantidad de fibras que había en el lado de entrada. Cada fibra de salida contiene un núcleo corto construido de manera especial, que filtra todas las longitudes de onda excepto una. Las señales resultantes se pueden encaminar a su destino o se pueden recombinar de distintas formas para un transporte multiplexado adicional.

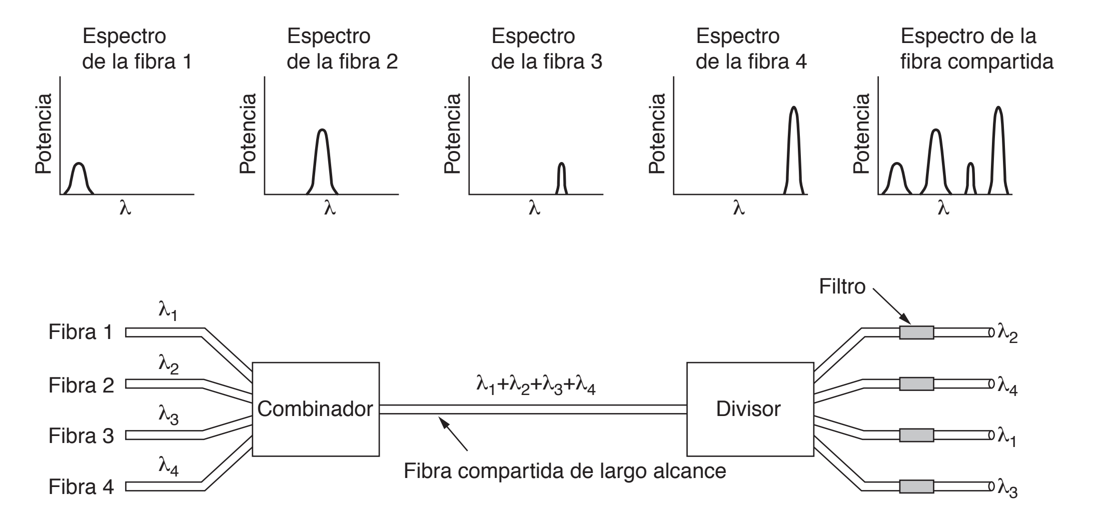


Figura 2-41. Multiplexión por división de longitud de onda.

En realidad aquí no hay nada nuevo. Esta forma de operar es simplemente una multiplexión por división de frecuencia a frecuencias muy altas, y el término WDM se debe a la descripción de los canales de fibra óptica con base en su longitud de onda o "color", en vez de la frecuencia. Mientras que cada canal tenga su propio rango de frecuencias (longitud de onda) y todos los intervalos estén separados, se pueden multiplexar juntos en la fibra de largo alcance. La única diferencia con la FDM eléctrica es que un sistema óptico que usa una rejilla de difracción es completamente pasivo y, por ende, muy confiable.

La razón por la que la WDM es popular, es que la energía en un solo canal por lo general es de sólo unos cuantos gigahertz, ya que ése es el límite actual de la rapidez con la que podemos realizar conversiones entre las señales eléctricas y ópticas. Al operar muchos canales en paralelo en distintas longitudes de onda, el ancho de banda agregado se incrementa en forma lineal con el número de canales. Como el ancho de banda de una sola banda de fibra es de alrededor de 25 000 GHz (vea la figura 2-7), en teoría hay espacio para 2 500 canales de 10 Gbps incluso a 1 bit/Hz (y también es posible obtener tasas de transmisión más altas).

La tecnología de la WDM ha estado progresando a una tasa que pone en vergüenza a la tecnología de las computadoras. La WDM se inventó alrededor del año 1990. Los primeros sistemas comerciales tenían ocho canales de 2.5 Gbps por canal. Para 1998 había en el mercado sistemas con 40 canales de 2.5 Gbps. Para 2006 había productos con 192 canales de 10 Gbps y 64 canales de 40 Gbps, capaces de desplazar hasta 2.56 Tbps. Este ancho de banda es lo bastante amplio como para transmitir 80 películas completas de DVD por segundo. Además, los canales están bien empaquetados en la fibra, con 200, 100 o incluso hasta 50 GHz de separación. Las demostraciones tecnológicas de las empresas después de sus alardeos han demostrado 10 veces esta capacidad en el laboratorio, pero para pasar del laboratorio al campo por lo general se requieren unos cuantos años como mínimo. Cuando hay una gran cantidad de canales y las longitudes de onda están muy cerca unas de otras, el sistema se llama **DWDM** (**WDM densa**, del inglés *Dense WDM*).

Uno de los factores cruciales de la tecnología WDM es el desarrollo de componentes totalmente ópticos. Antes era necesario dividir cada 100 km los canales y convertir cada uno de ellos en una señal eléctrica para poder amplificarlos por separado antes de volver a convertirlos en señales ópticas y combinarlos. En la actualidad, los amplificadores totalmente ópticos pueden regenerar la señal completa una vez cada 1000 km sin necesidad de realizar múltiples conversiones opto-eléctricas.

En el ejemplo de la figura 2-41 tenemos un sistema de longitud de onda fija. Los bits de la fibra 1 de entrada van a la fibra 3 de salida, los bits de la fibra 2 de entrada van a la fibra 1 de salida, etc. Aunque también es posible construir sistemas WDM que se conmuten en el dominio óptico. En dicho dispositivo, los filtros de salida se ajustan mediante interferómetros Fabry-Perot o Mach-Zehnder. Mediante estos dispositivos, una computadora de control puede cambiar las frecuencias seleccionadas en forma dinámica. Esta habilidad proporciona una gran cantidad de flexibilidad para proporcionar muchas trayectorias de longitud de onda distintas a través de la red telefónica, a partir de un conjunto fijo de fibras. Para obtener información sobre las redes ópticas y WDM, consulte a Ramaswami y colaboradores (2009).

2.6.5 Conmutación

Desde el punto de vista de un ingeniero de telefonía ordinario, el sistema telefónico se divide en dos partes principales: la planta externa (lazos locales y troncales, ya que están físicamente fuera de las oficinas de conmutación) y la planta interna (los conmutadores, que están dentro de las oficinas de conmutación). Ya vimos la planta externa. Ahora es tiempo de examinar la planta interna.

Hoy en día se utilizan dos técnicas de conmutación diferentes en las redes: conmutación de circuitos y conmutación de paquetes. El sistema telefónico tradicional está basado en la conmutación de circuitos,

aunque la conmutación de paquetes está empezando a tener avances significativos con el surgimiento de la tecnología de voz sobre IP. Analizaremos con cierto detalle la conmutación de circuitos y la contrastaremos con la conmutación de paquetes. Ambos tipos de conmutación son muy importantes, así que volveremos a hablar sobre ellos cuando veamos la capa de red.

Conmutación de circuitos

En teoría, cuando usted o su computadora hacen una llamada telefónica, el equipo de conmutación dentro del sistema telefónico busca una trayectoria física desde su teléfono hasta el teléfono del receptor. Esta técnica se conoce como **conmutación de circuitos**. En la figura 2-42(a) se muestra un esquema de ella. Cada uno de los seis rectángulos representa a una oficina de conmutación de la portadora (oficina final, oficina interurbana, etc.). En este ejemplo, cada oficina tiene tres líneas entrantes y tres líneas salientes. Cuando una llamada pasa a través de una oficina de conmutación, se establece una conexión física (en forma conceptual) entre la línea por la que llegó la llamada y una de las líneas de salida, como se indica mediante las líneas punteadas.

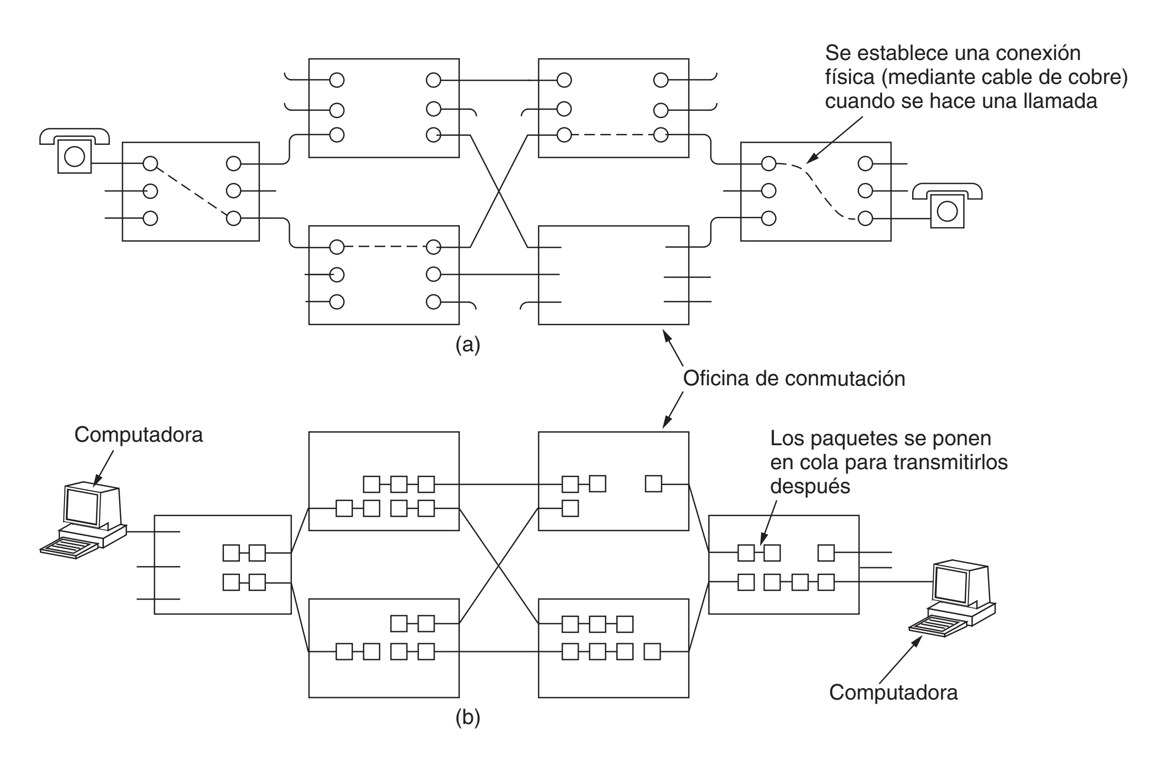


Figura 2-42. (a) Conmutación de circuitos. (b) Conmutación de paquetes.

En los primeros días del teléfono, la operadora establecía la conexión al conectar un cable de puenteo en los enchufes de entrada y salida. De hecho, hay una pequeña y sorprendente historia asociada con la invención del equipo de conmutación automática de circuitos, inventado en el siglo XIX por el dueño de una funeraria de Missouri, llamado Almon B. Strowger. Poco después de la invención del teléfono, cuando alguien moría, uno de los que le sobrevivían llamaba a la operadora del pueblo y decía: “Por favor comuníqueme con una funeraria”. Por desgracia para el señor Strowger, había dos funerarias en el pueblo y la esposa del dueño de la otra era la operadora telefónica de esa comunidad. Strowger pronto se dio cuenta de

que si no inventaba un equipo de conmutación telefónica automática iba a quedar en bancarrota, así que eligió la primera opción. Durante casi 100 años, el equipo de conmutación de circuitos empleado en todo el mundo se conoció como el **aparato de Strowger** (la historia no registra si la ahora desempleada operadora de conmutador obtuvo un empleo como operadora de información, contestando preguntas como: “¿Me puede proporcionar el número telefónico de una funeraria?”).

Cabe mencionar que el modelo que se muestra en la figura 2-42(a) está muy simplificado, ya que partes de la trayectoria física entre los dos teléfonos pueden ser, de hecho, enlaces de microondas o de fibra óptica en los cuales se multiplexan miles de llamadas. Sin embargo, la idea básica es válida: una vez que se establece una llamada, existe una trayectoria dedicada entre ambos extremos y seguirá existiendo hasta que termine la llamada.

Una propiedad importante de la conmutación de circuitos es la necesidad de establecer una trayectoria de extremo a extremo *antes* de poder enviar datos. El tiempo transcurrido desde el fin de la marcación hasta que el timbre empieza a sonar puede ser fácilmente de 10 seg, y puede ser aún mayor en las llamadas de larga distancia o internacionales. Durante este intervalo, el sistema telefónico está buscando una trayectoria, como se muestra en la figura 2-43(a). Tenga en cuenta que, antes de que pueda siquiera empezar la transmisión de datos, la señal de petición de llamada se debe propagar hasta el destino y se debe confirmar. En muchas aplicaciones de computadora (por ejemplo, la verificación de crédito en un punto de venta), no son deseables los tiempos de establecimiento largos.

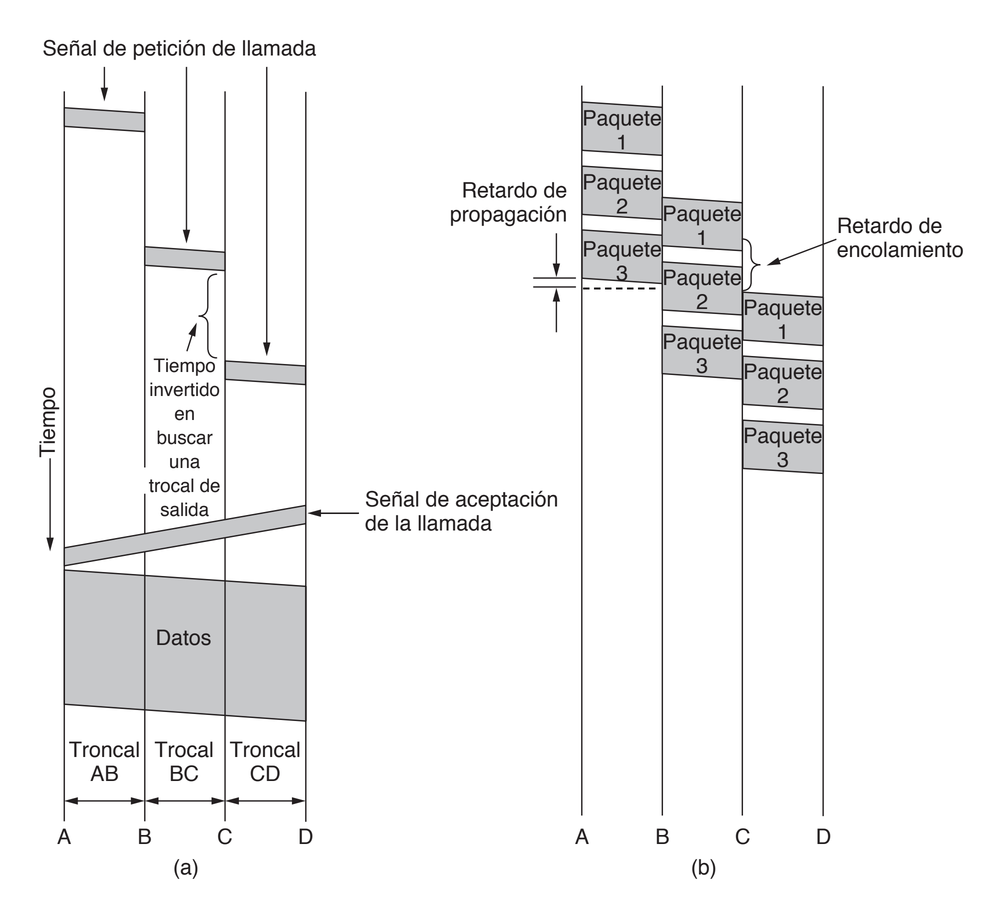


Figura 2-43. Sincronización de los eventos en (a) conmutación de circuitos, (b) conmutación de paquetes.

Como consecuencia de la trayectoria reservada entre las partes que participan en la llamada, una vez que se completa el establecimiento, el único retardo de los datos es el tiempo de propagación de la señal electromagnética, alrededor de 5 mseg por cada 1000 km. Además, como consecuencia de la trayectoria establecida, no hay peligro de congestión; es decir, una vez que entra la llamada, no hay posibilidad de obtener señales de ocupado. Claro que podría obtener una antes de establecer la conexión debido a la falta de la capacidad de conmutación o de troncal.

Conmutación de paquetes

La alternativa a la conmutación de circuitos se llama **conmutación de paquetes**, que se muestra en la figura 2-42(b) y la cual describimos en el capítulo 1. Con esta tecnología, los paquetes se envían tan pronto como están disponibles. No hay necesidad de establecer una ruta dedicada de antemano, a diferencia de la conmutación de circuitos. Los enrutadores usan transmisión de almacenamiento y envío para enviar cada paquete a su destino por separado. Este procedimiento es distinto a la conmutación de circuitos, en la cual el resultado del establecimiento de la conexión es que se reserva todo el ancho de banda desde el emisor hasta el receptor. Todos los datos en el circuito siguen esta trayectoria. Entre otras propiedades, hacer que todos los datos sigan la misma ruta significa que no pueden llegar desordenados. En la conmutación de paquetes no hay una trayectoria fija, por lo que los distintos paquetes pueden seguir diferentes trayectorias, dependiendo de las condiciones de red al momento en que se enviaron, y por lo tanto pueden llegar desordenados.

Las redes de conmutación de paquetes imponen un límite superior estrecho sobre el tamaño de los paquetes. Esto asegura que ningún usuario pueda monopolizar una línea de transmisión por mucho tiempo (es decir, muchos milisegundos), de modo que las redes de conmutación de paquetes pueden manejar tráfico interactivo. También se reduce el retardo, ya que el primer paquete de un mensaje largo se puede reenviar antes de que haya llegado el segundo por completo. Sin embargo, el retardo de almacenamiento y reenvío por acumular un paquete en la memoria del enrutador antes de enviarlo al siguiente enrutador es mayor que en la conmutación de circuitos. En la conmutación de circuitos, los bits simplemente fluyen por el cable en forma continua.

La conmutación de paquetes y la de circuitos también difieren en otros aspectos. Como en la conmutación de paquetes no se reserva ancho de banda, tal vez los paquetes tengan que esperar a ser reenviados. Esto introduce **retardo de encolamiento** y congestión si se envían muchos paquetes al mismo tiempo. Por otra parte, no hay peligro de obtener un tono de ocupado o de no poder usar la red. Así, la congestión ocurre en distintos tiempos en la conmutación de circuitos (al momento de establecer la llamada) y en la conmutación de paquetes (al momento de enviar los paquetes).

Si se reservó un circuito para un usuario específico y no hay tráfico, se desperdicia su ancho de banda. No se puede usar para otro tráfico. La conmutación de paquetes no desperdicia ancho de banda y, por ende, es más eficiente desde la perspectiva del sistema. Entender este compromiso es crucial para comprender la diferencia entre la conmutación de circuitos y la conmutación de paquetes. El compromiso está entre un servicio garantizado con desperdicio de recursos contra un servicio no garantizado pero sin desperdicio de recursos.

La conmutación de paquetes es más tolerante a errores que la conmutación de circuitos. De hecho, es la razón por la que se inventó. Si un conmutador falla, se terminan todos los circuitos que lo utilizan y no se puede enviar más tráfico a través de ninguno de ellos. En la conmutación de paquetes, se pueden encaminar paquetes alrededor de interruptores inhabilitados.

Una diferencia final entre la conmutación de circuitos y la de paquetes es el algoritmo de cobro. En la conmutación de circuitos, el cobro se ha basado históricamente en la distancia y el tiempo. Por lo general, para los teléfonos móviles la distancia no es importante, excepto en las llamadas internacionales, y el tiempo sólo desempeña un papel general (por ejemplo, un plan de llamadas con 2 000 minutos gratis cuesta más que uno con 1000 minutos gratis y algunas veces las noches o los fines de semana son económicos). En la conmutación de paquetes el tiempo de conexión no representa un problema, pero el volumen

Elemento	Conmutación de circuitos	Conmutación de paquetes
Establecimiento de llamadas.	Requerido	No es necesario.
Trayectoria física dedicada.	Sí	No
Cada paquete sigue la misma trayectoria.	Sí	No
Los paquetes llegan en orden.	Sí	No
Una falla en un conmutador es fatal.	Sí	No
Ancho de banda disponible.	Fijo	Dinámico.
Tiempo de una posible congestión.	Durante el establecimiento de la llamada.	En todos los paquetes.
Ancho de banda potencialmente desperdiciado.	Sí	No
Transmisión de almacenamiento y envío.	No	Sí
Cobro.	Por minuto.	Por paquete.

Figura 2-44. Comparación de las redes de conmutación de circuitos y conmutación de paquetes.

de tráfico sí. Para los usuarios domésticos, por lo general los ISP cobran una cuota mensual fija debido a que es menos trabajo para ellos y sus clientes pueden comprender este modelo, pero las portadoras de red troncal cobran a las redes regionales con base en el volumen de su tráfico.

En la figura 2-44 se resumen las diferencias. Las redes telefónicas han utilizado tradicionalmente la conmutación de circuitos para proveer llamadas telefónicas de alta calidad, y las redes de computadoras han usado la conmutación de paquetes por cuestión de simpleza y eficiencia. Sin embargo, existen excepciones notables. Algunas redes de computadoras antiguas han usado la conmutación de circuitos (por ejemplo, X25) y algunas redes telefónicas más recientes usan conmutación de paquetes con tecnología de voz sobre IP. Por fuera se ve como una llamada telefónica estándar para los usuarios, pero en su interior los paquetes de red que contienen los datos de voz son conmutados. Esta propuesta ha permitido a las nuevas compañías comercializar llamadas internacionales económicas por medio de las tarjetas de llamadas, aunque tal vez con una menor calidad en la llamada en comparación con las compañías establecidas.

2.7 El SISTEMA DE TELEFONÍA MÓVIL

Aun si el sistema de telefonía tradicional llegara algún día a utilizar fibra óptica con capacidad de multigigabit desde un extremo hasta el otro, no podría satisfacer a un grupo creciente de usuarios: las personas en movimiento. Ahora las personas esperan hacer llamadas telefónicas y usar sus teléfonos para revisar su correo electrónico y navegar por la web desde aviones, autos, albercas y mientras trotan en el parque. En consecuencia, hay mucho interés en la telefonía inalámbrica. En las siguientes secciones estudiaremos este tema con detalle.

El sistema de telefonía móvil se utiliza para la comunicación de datos y voz de área amplia. Los **teléfonos móviles** (también conocidos como **teléfonos celulares**) han pasado por tres generaciones distintas, conocidas comúnmente como **1G**, **2G** y **3G**. Las generaciones son:

1. Voz analógica.
2. Voz digital.
3. Voz y datos digitales (Internet, correo electrónico, etcétera).

(No debemos confundir los teléfonos móviles con los **teléfonos inalámbricos**, los cuales consisten en una estación base y un auricular que se venden en conjunto para usarlos dentro del hogar. Éstos nunca se utilizan para las redes, de modo que no los examinaremos con más detalle).

Aunque la mayor parte de nuestro estudio se enfoca en la tecnología de estos sistemas, es interesante observar cómo las decisiones políticas y de mercado pueden tener un enorme impacto. El primer sistema móvil fue inventado en Estados Unidos por AT&T y se hizo obligatorio en todo el país mediante la FCC. Como resultado, todo Estados Unidos tenía un solo sistema (analógico), por lo que un teléfono móvil comprado en California también funcionaba en Nueva York. En contraste, cuando los teléfonos móviles llegaron a Europa cada país desarrolló su propio sistema, lo cual resultó en un fiasco.

Europa aprendió de su error y cuando llegó el sistema digital, las PTT gubernamentales se reunieron y crearon estándares para un solo sistema (GSM), de modo que cualquier teléfono móvil europeo pudiera funcionar en cualquier parte de Europa. Para entonces, Estados Unidos había decidido que el gobierno no debería participar en el negocio de la estandarización, así que dejó el sistema digital a cargo del mercado. Debido a esta decisión, los distintos fabricantes de equipos produjeron tipos diferentes de teléfonos móviles. Como consecuencia, en Estados Unidos se implementaron dos tipos principales (y totalmente incompatibles) de sistemas de telefonía móvil digital, así como algunos otros sistemas menos importantes.

A pesar del liderazgo inicial de Estados Unidos, ahora hay más propietarios y usuarios de teléfonos móviles en Europa. Tener un solo sistema que funcione en cualquier parte de Europa y con cualquier proveedor es parte de la razón, pero hay algo más. Una segunda área en la que difieren Estados Unidos y Europa es en la humilde cuestión de los números telefónicos. En Estados Unidos, los teléfonos móviles están mezclados con los teléfonos comunes (fijos). Por ende, no hay forma de que la persona que va a hacer una llamada pueda ver si, por ejemplo, el número (212) 234-5678 es fijo (llamada económica o gratuita) o si es teléfono móvil (llamada costosa). Para evitar que las personas se pongan nerviosas a la hora de hacer llamadas, las compañías telefónicas decidieron hacer que el propietario del teléfono móvil pague por las llamadas entrantes. Como consecuencia, muchas personas dudaban al comprar un teléfono celular por miedo a generar una factura extensa sólo por recibir llamadas. En Europa, los números de teléfonos móviles tienen un código de área especial (algo parecido a los números 800 y 900) para reconocerlos al instante. Por lo tanto, la regla usual "el que llama paga" se aplica también a los teléfonos móviles en Europa (excepto las llamadas internacionales, en las que se dividen los costos).

Un tercer aspecto que ha tenido un gran impacto sobre la adopción de los teléfonos móviles es el extenso uso de los teléfonos móviles prepagados en Europa (hasta un 75% en algunas áreas). Estos teléfonos se pueden comprar en muchas tiendas con la misma formalidad como si se comprara una cámara digital. Usted sólo paga y se va. Vienen precargados con un saldo de, por ejemplo, 20 o 50 euros, y se pueden recargar (mediante un código NIP secreto) cuando el saldo queda en cero. Debido a esto, prácticamente todos los adolescentes y muchos niños pequeños en Europa tienen teléfonos móviles (por lo general, prepagados) para que sus padres puedan localizarlos, sin el peligro de que el niño pague una enorme cuenta, ya que no hay un cargo mensual por las llamadas entrantes.

2.7.1 Teléfonos móviles de primera generación (1G): voz analógica

Dejemos a un lado los aspectos políticos y de marketing de los teléfonos móviles. Ahora analizaremos la tecnología, empezando con el primero de los sistemas. Los radioteléfonos móviles se utilizaban esporádicamente para comunicación marítima y militar durante las primeras décadas del siglo XX. En 1946 se estableció el primer sistema de teléfonos instalados en autos en St. Louis. Este sistema utilizaba un solo transmisor grande colocado en la parte superior de un edificio alto y tenía un solo canal, el cual servía tanto para enviar como para recibir. Para hablar, el usuario tenía que oprimir un botón para habilitar el transmisor y deshabilitar el receptor. Dichos sistemas, conocidos como **sistemas de oprimir para**

hablar, se instalaron en varias ciudades desde finales de la década de 1950. El radio de banda civil (CB), los taxis y las patrullas utilizan con frecuencia esta tecnología.

En la década de 1960, se instaló el **IMTS** (**Sistema Mejorado de Telefonía Móvil**, del inglés *Improved Mobile Telephone System*). También utilizaba un transmisor de alta potencia (200 watts) en la parte superior de una colina, pero tenía dos frecuencias: una para enviar y otra para recibir. De esta forma, ya no era necesario el botón de oprimir para hablar. Como toda la comunicación de los teléfonos móviles entraba por un canal distinto al de las señales de salida, los usuarios móviles no se podían escuchar entre sí (a diferencia del sistema de oprimir para hablar que utilizaban los taxis).

El IMTS manejaba 23 canales dispersos desde 150 MHz hasta 450 MHz. Debido al pequeño número de canales, a veces los usuarios tenían que esperar mucho tiempo antes de obtener el tono de marcar. Además, debido a la gran potencia de los transmisores en la cima de las colinas, los sistemas adyacentes tenían que estar separados varios cientos de kilómetros para evitar interferencia. En sí, el sistema no era práctico debido a su limitada capacidad.

Sistema avanzado de telefonía móvil

Todo cambió con el **AMPS** (**Sistema Avanzado de Telefonía Móvil**, del inglés *Advanced Mobile Phone System*), inventado por los Laboratorios Bell e instalado por primera vez en Estados Unidos en 1982. También se utilizó en Inglaterra, donde se llamaba TACS, y en Japón, donde se llamaba MCS-L1. El AMPS se retiró formalmente en 2008, pero lo analizaremos para comprender el contexto de los sistemas 2G y 3G que se basaron en él para mejorar.

En todos los sistemas de telefonía móvil, una región geográfica se divide en **celdas**, razón por la cual a los dispositivos se les conoce comúnmente como teléfonos celulares. En AMPS, las celdas tienen normalmente un alcance de de 10 a 20 km; en los sistemas digitales las celdas son más pequeñas. Cada celda utiliza cierto conjunto de frecuencias que no utiliza ninguna de las celdas adyacentes. La idea clave que confiere a los sistemas celulares una mayor capacidad que la de los sistemas anteriores es el uso de celdas relativamente pequeñas y la reutilización de las frecuencias de transmisión en celdas cercanas (pero no adyacentes). Mientras que un sistema IMTS de 100 km de alcance sólo puede tener una llamada en cada frecuencia, un sistema AMPS podría tener 100 celdas de 10 km en la misma área y ser capaz de tener de 10 a 15 llamadas en cada frecuencia, en celdas muy separadas. Así, el diseño celular incrementa la capacidad del sistema cuando menos en un orden de magnitud, o más si las celdas son más pequeñas. Asimismo, tener celdas más pequeñas significa que se requiere menos potencia, lo cual nos permite usar transmisores y auriculares más pequeños y baratos.

La idea de reutilizar la frecuencia se ilustra en la figura 2-45(a). Por lo general las celdas son casi circulares, pero es más fácil modelarlas como hexágonos. En la figura 2-45(a), todas las celdas son del mismo tamaño. Están agrupadas en unidades de siete celdas. Cada letra indica un grupo de frecuencias. Hay que tener en cuenta que para cada conjunto de frecuencias, hay un espacio de aproximadamente dos celdas de ancho en donde no se reutiliza esa frecuencia para tener una buena separación y una interferencia baja.

Encontrar ubicaciones elevadas para colocar antenas de estación base es un gran problema, el cual ha provocado que algunas portadoras de telecomunicaciones formen alianzas con la Iglesia Católica Romana, ya que ésta posee una cantidad considerable de sitios potenciales para antenas en todo el mundo, los cuales se encuentran convenientemente bajo la misma administración.

En un área en la que el número de usuarios ha crecido a tal extremo que el sistema está sobrecargado, se puede reducir la potencia y las celdas sobrecargadas se dividen en **microceldas** más pequeñas para permitir una mayor reutilización de frecuencias, como se muestra en la figura 2-45(b). Algunas veces las compañías telefónicas crean microceldas temporales mediante el uso de torres portátiles con enlaces de satélite en eventos deportivos, conciertos de rock y otros lugares en donde se congregan grandes cantidades de usuarios móviles por unas cuantas horas.

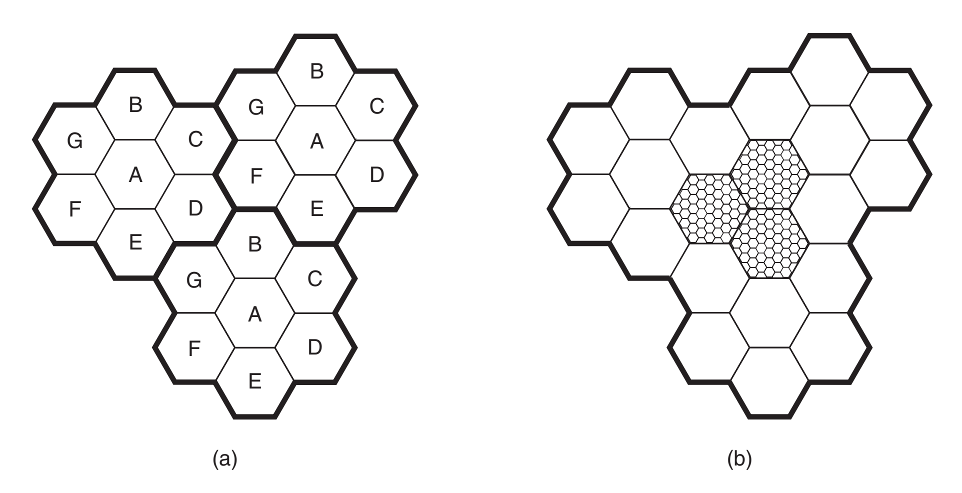


Figura 2-45. (a) Las frecuencias no se reutilizan en las celdas adyacentes. (b) Para agregar más usuarios se pueden usar celdas más pequeñas.

En el centro de cada celda hay una estación base a la cual transmiten todos los teléfonos de la celda. La estación base consiste en una computadora y un transmisor/receptor conectado a una antena. En un sistema pequeño, todas las estaciones base se conectan a un solo dispositivo llamado **MSC** (**Centro de Conmutación Móvil**, del inglés *Mobile Switching Center*) o **MTSO** (**Oficina de Conmutación de Telefonía Móvil**, del inglés *Mobile Telephone Switching Office*). En un sistema más grande pueden ser necesarios más MSC, los cuales se conectan a un MSC de segundo nivel, y así en lo sucesivo. En esencia los MSC son oficinas finales como en el sistema telefónico y, de hecho, están conectados cuando menos a una oficina final del sistema telefónico. Los MSC se comunican con las estaciones base, con otros MSC y con la PSTN mediante el uso de una red de conmutación de paquetes.

En cualquier instante, cada teléfono móvil está lógicamente en una celda específica y bajo el control de la estación base de esa celda. Cuando un teléfono móvil sale físicamente de una celda, su estación base detecta que la señal telefónica se desvanece y pregunta a todas las estaciones base circundantes cuánta potencia están recibiendo de ese teléfono. Al recibir las respuestas, la estación base transfiere la propiedad a la celda que está recibiendo la señal más fuerte; en la mayoría de las condiciones es la celda en la que se encuentra el teléfono en ese momento. A continuación se informa al teléfono sobre su nuevo jefe y, si hay una llamada en progreso, se le pide que cambie a un nuevo canal (puesto que el anterior no se reutiliza en ninguna de las celdas adyacentes). Este proceso se conoce como **entrega** (*handoff*) y tarda cerca de 300 mseg. El MSC (el centro neurálgico del sistema) se encarga de asignar el canal. Las estaciones base en realidad sólo son retransmisoras de radio sin inteligencia.

Canales

El sistema AMPS utiliza FDM para separar los canales. Emplea 832 canales full-dúplex, cada uno de los cuales consiste en un par de canales simplex. Este arreglo se conoce como **FDD** (**Duplexión por División de Frecuencia**, del inglés *Frequency Division Duplex*). Los 832 canales simplex de 824 a 849 MHz se utilizan para la transmisión del móvil a la estación base, y los 832 canales simplex de 869 a 894 MHz se utilizan para transmisiones de la estación base al móvil. Cada uno de estos canales simplex tiene una amplitud de 30 kHz.

Los 832 canales se dividen en cuatro categorías. Canales de control (base a móvil) se utilizan para administrar el sistema. Canales de localización (base a móvil) alertan a los usuarios móviles que tienen

llamadas. Canales de acceso (bidireccional) se usan para establecimiento de llamadas y asignación de canales. Por último, los canales de datos (bidireccional) transportan voz, fax o datos. Puesto que no se pueden reutilizar las mismas frecuencias en las celdas cercanas y se reservan 21 canales en cada celda para actividades de control, el número actual de canales de voz disponibles por celda es mucho menor de 832, por lo general cerca de 45.

Administración de llamadas

En AMPS, cada teléfono móvil tiene un número de serie de 32 bits y un número telefónico de 10 dígitos en su memoria programable de sólo lectura. El número telefónico se representa como un código de área de tres dígitos en 10 bits y un número de suscriptor de 7 dígitos en 24 bits. Al encender un teléfono, examina una lista preprogramada de 21 canales de control para encontrar la señal más potente. Después el teléfono difunde su número de serie de 32 bits y su número telefónico de 34 bits. Al igual que toda la información de control en el AMPS, este paquete se envía en formato digital varias veces y con un código de corrección de errores, aun cuando los canales de voz en sí son analógicos.

Cuando la estación base escucha el anuncio, avisa al MSC y éste registra la existencia de su nuevo cliente además de informar al MSC local del cliente sobre su ubicación actual. Durante la operación normal el teléfono móvil se vuelve a registrar cada 15 minutos aproximadamente.

Para hacer una llamada, el usuario enciende el teléfono móvil, teclea el número al que va a llamar y oprime el botón ENVIAR (*Send*). El teléfono entonces transmite el número a llamar y su propia identidad por el canal de acceso. Si ocurre una colisión en este punto, vuelve a intentar más tarde. Cuando la estación base recibe la petición, informa al MSC. Si el que llama es cliente de la compañía del MSC (o de uno de sus socios), el MSC busca un canal inactivo para la llamada. Si encuentra uno, se envía el número de ese canal de vuelta por el canal de control. A continuación el teléfono móvil conmuta de manera automática al canal de voz seleccionado y espera a que la persona a la que llamó levante el teléfono.

Las llamadas entrantes funcionan en forma distinta. Para empezar, todos los teléfonos inactivos escuchan de manera continua el canal de localización para detectar los mensajes dirigidos a ellos. Cuando se hace una llamada a un teléfono móvil (ya sea desde un teléfono fijo o de otro teléfono móvil), se envía un paquete al MSC local del que va a recibir la llamada para averiguar su ubicación. Después se envía un paquete a la estación base de su celda actual, que envía una difusión por el canal de localización de la forma: “Unidad 14, ¿está ahí?”. El teléfono al que se llamó responde “Sí” por el canal de acceso. Después la base dice algo así como: “Unidad 14, llamada para usted por el canal 3”. En este punto, el teléfono al que se llamó conmuta al canal 3 y empieza a timbrar (o reproduce alguna melodía que recibió el propietario de regalo).

2.7.2 Teléfonos móviles de segunda generación (2G): voz digital

La primera generación de teléfonos móviles era analógica; la segunda, es digital. El cambio a digital tiene varias ventajas. Ofrece un aumento en la capacidad al permitir la digitalización y compresión de las señales de voz. Mejora la seguridad al permitir cifrar las señales de voz y de control. Esto a su vez impide los fraudes y el espionaje, ya sean producto de una exploración intencional o debido a los ecos de otras llamadas que se producen por la propagación de RF. Por último, permite el uso de nuevos servicios, como la mensajería de texto.

Así como en la primera generación no hubo una estandarización a nivel mundial, tampoco la segunda cuenta con ello. Se desarrollaron varios sistemas distintos, de los cuales tres se han implementado ampliamente. **D-AMPS** (**Sistema Avanzado de Telefonía Móvil Digital**, del inglés *Digital Advanced Mobile Phone System*) es una versión digital de AMPS que coexiste con este sistema y usa TDM para colocar múltiples llamadas en el mismo canal de frecuencia. Se describe en el estándar internacional IS-54 y en

su sucesor, IS-136. **GSM** (**Sistema Global para Comunicaciones Móviles**, del inglés *Global System for Mobile communications*) se ha establecido como el sistema dominante, y aunque tardó en popularizarse en Estados Unidos ahora se utiliza casi en cualquier parte del mundo. Al igual que D-AMPS, GSM se basa en una mezcla de FDM y TDM. El sistema **CDMA** (**Acceso Múltiple por División de Código**, del inglés *Code Division Multiple Access*), que se describe en el **estándar internacional IS-95**, es un tipo de sistema completamente distinto y no se basa en FDM ni en TDM. Aunque CDMA no se ha convertido en el sistema 2G dominante, su tecnología forma la base para los sistemas 3G.

Algunas veces se utiliza también el nombre **PCS** (**Servicios de Comunicaciones Personales**, del inglés *Personal Communications Services*) en la literatura para indicar un sistema de segunda generación (es decir, digital). En un principio indicaba que un teléfono móvil usaba la banda de 1900 MHz, pero en la actualidad es raro hacer esta distinción.

Ahora describiremos el sistema GSM, puesto que es el sistema 2G dominante. En la siguiente sección hablaremos más sobre CDMA cuando describamos los sistemas 3G.

GSM – El sistema global para comunicaciones móviles

GSM empezó en la década de 1980 como un esfuerzo por producir un solo estándar europeo de 2G. La tarea se asignó a un grupo de telecomunicaciones llamado (en francés) Groupe Specialé Mobile. Los primeros sistemas GSM se implementaron a partir de 1991 y fueron un éxito inmediato. Pronto quedó claro que GSM iba a ser más que un éxito en Europa, puesto que se estaba popularizando en países tan lejanos como Australia, por lo cual se cambió su nombre para que tuviera un atractivo más enfocado al mercado mundial.

Al igual que los sistemas 1G, GSM y los demás sistemas de telefonía móvil que estudiaremos conservan un diseño basado en celdas, la reutilización de frecuencias entre celdas y la movilidad mediante entregas, a medida que se mueven los suscriptores. Son los detalles los que difieren. Aquí analizaremos brevemente algunas de las propiedades principales del sistema GSM. No obstante, el estándar impreso de GSM es de más de 5 000 [*sic*] páginas. Una gran parte de este material se relaciona con los aspectos de ingeniería del sistema, en especial el diseño de los receptores para manejar la propagación de señales multitrayectorias y la sincronización de los transmisores y receptores. Aquí ni siquiera mencionaremos algo de esto.

La figura 2-46 muestra que la arquitectura GSM es similar a la arquitectura AMPS, aunque los componentes tienen distintos nombres. Ahora el móvil en sí es dividido en el teléfono y en un chip removible con información del suscriptor y la cuenta, conocido como **tarjeta SIM** (**Módulo de Identidad del Suscriptor**, del inglés *Subscriber Identity Module*). La tarjeta SIM activa el teléfono y contiene los secretos que permiten al móvil y a la red identificarse entre sí y cifrar las conversaciones. Es posible quitar la tarjeta SIM e insertarla en un teléfono distinto para convertirlo en su teléfono móvil en lo que respecta a la red.

El teléfono móvil habla con las estaciones base celulares a través de una **interfaz aérea** que describiremos en un momento. Cada una de las estaciones base celulares están conectadas a un **BSC** (**Controlador**

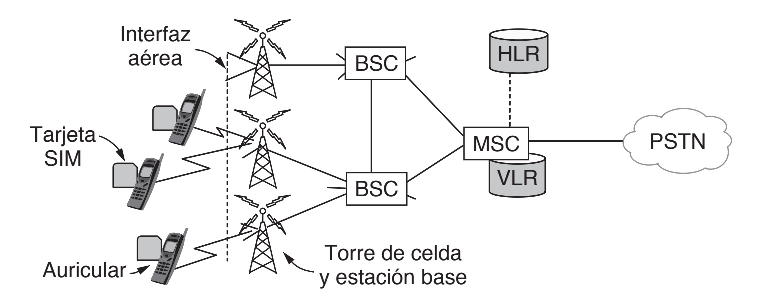


Figura 2-46. Arquitectura de la red de telefonía GSM.

de Estación Base, del inglés *Base Station Controller*) que controla los recursos de radio de las celdas y maneja la entrega. A su vez, el BSC está conectado a un MSC (como en el sistema AMPS) que encamina las llamadas y se conecta a la PSTN (Red Telefónica Pública Conmutada, del inglés *Public Switched Telephone Network*).

Para encaminar las llamadas, el MSC necesita saber en dónde puede encontrar los móviles en un momento dado. Para ello mantiene una base de datos de los móviles cercanos asociados con la celda que administra. A esta base de datos se le conoce como **VLR** (**Registro de Ubicación de Visitante**, del inglés *Visitor Location Register*). También hay una base de datos en la red móvil que proporciona la última ubicación conocida de cada móvil y se llama **HLR** (**Registro de Ubicación Local**, del inglés *Home Location Register*). Esta base de datos se utiliza para encaminar las llamadas entrantes a las ubicaciones correctas. Ambas bases de datos se deben mantener actualizadas a medida que los teléfonos móviles se desplazan de una celda a otra.

Ahora describiremos la interfaz aérea con cierto detalle. GSM opera en un rango de frecuencias a nivel mundial, incluyendo 900, 1 800 y 1 900 MHz. Se asigna más espectro que en el AMPS para soportar a un número mucho mayor de usuarios. Al igual que AMPS, GSM es un sistema celular por división de frecuencia dúplex. Esto es, cada móvil transmite en una frecuencia y recibe en otra más alta (55 MHz más alta para GSM y 80 MHz más alta para AMPS). Pero a diferencia de AMPS, en GSM un solo par de frecuencias se divide mediante multiplexión por división de tiempo en ranuras de tiempo. De esta forma se puede compartir entre múltiples móviles.

Para manejar múltiples móviles, los canales GSM son mucho más amplios que los canales AMPS (200 kHz, en comparación con 30 kHz). En la figura 2-47 se muestra un canal de 200 kHz. Un sistema GSM que opera en la región de 900 MHz tiene 124 pares de canales simplex. Cada canal simplex tiene una amplitud de 200 kHz y soporta ocho conexiones separadas mediante el uso de multiplexión por división de tiempo. Cada estación activa en un momento dado recibe una ranura de tiempo por cada par de canales. En teoría cada celda puede soportar 992 canales, pero muchos de ellos no están disponibles para evitar conflictos de frecuencias con las celdas adyacentes. En la figura 2-47, las ocho ranuras de tiempo sombreadas pertenecen a la misma conexión, cuatro de ellos en cada sentido. La transmisión y la recepción no se llevan a cabo en la misma ranura de tiempo debido a que los radios GSM no pueden recibir y transmitir a la vez, además de que se requiere cierto tiempo para conmutar entre uno y otro. Si el dispositivo móvil asignado a 890.4/935.4 MHz y a la ranura de tiempo 2 quisiera transmitir a la estación base, usaría las cuatros ranuras inferiores sombreadas (y las que les siguen en el tiempo), colocando algunos datos en cada ranura hasta terminar de enviar todos los datos.

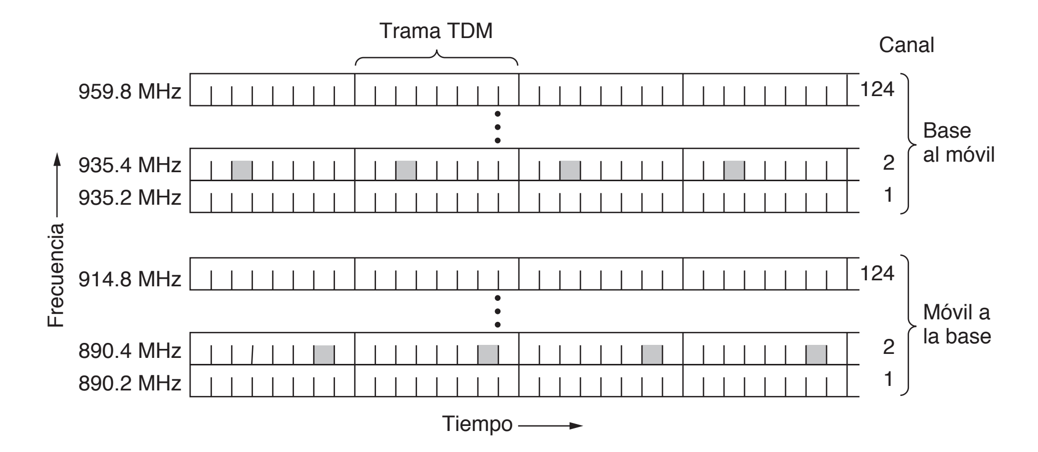


Figura 2-47. El sistema GSM utiliza 124 canales de frecuencia, cada uno de los cuales usa un sistema TDM de ocho ranuras.

Las ranuras TDM que se muestran en la figura 2-47 forman parte de una compleja jerarquía de entramado. Cada ranura TDM tiene una estructura específica; además los grupos de ranuras TDM forman multitramas, también con una estructura específica. En la figura 2-48 se muestra una versión simplificada de esta jerarquía. Aquí podemos ver que cada ranura TDM consiste en una trama de datos de 148 bits que ocupa el canal durante 577 μseg (incluyendo un tiempo de guarda de 30 μseg después de cada ranura). Cada trama de datos inicia y termina con tres bits 0, para fines de delinear las tramas. También contiene dos campos de *información* de 57 bits, cada uno de los cuales tiene un bit de control que indica si el siguiente campo de *información* es para voz o para datos. Entre los campos de *información* hay un campo de *sincronización* (entrenamiento) de 26 bits que el receptor utiliza para sincronizarse con los límites de la trama del emisor.

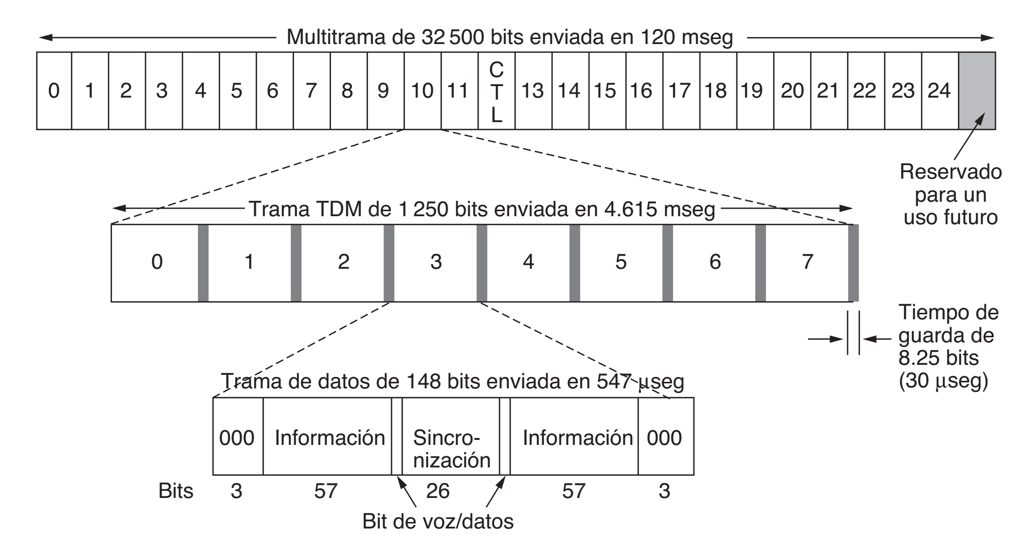


Figura 2-48. Una parte de la estructura de entramado de GSM.

Una trama de datos se transmite en 547 μseg, pero un transmisor sólo puede enviar una trama de datos cada 4.615 mseg debido a que está compartiendo el canal con otras siete estaciones. La tasa de transmisión bruta de cada canal es de 270,833 bps, dividida entre ocho usuarios. No obstante, al igual que en el sistema AMPS, la información adicional ocupa una gran parte del ancho de banda, lo cual nos deja en última instancia con 24.7 kbps de carga útil por usuario antes de la corrección de errores. Después de la corrección de errores quedan 13 kbps para la voz. Aunque esto es considerablemente menor que los 64 kbps PCM para las señales de voz sin compresión en la red de telefonía fija, la compresión en el dispositivo móvil puede llegar a estos niveles sin perder mucha calidad.

Como podemos ver de la figura 2-48, ocho tramas de datos conforman una trama TDM y 26 tramas TDM conforman una multitrama de 120 mseg. De las 26 tramas TDM en una multitrama, la ranura 12 se utiliza para fines de control y la ranura 25 está reservada para un uso futuro, por lo que sólo hay 24 disponibles para el tráfico de los usuarios.

Pero además de la multitrama de 26 ranuras que se muestra en la figura 2-48, también se utiliza una multitrama de 51 ranuras (no se muestra en la figura). Algunas de estas ranuras se utilizan para alojar varios canales de control con el objetivo de administrar el sistema. El **canal de control de difusión** es un flujo continuo de salida que proviene de la estación base y contiene la identidad de ésta junto con el estado del canal. Todas las estaciones móviles monitorean la potencia de su señal para ver si se han desplazado a una celda nueva.

El **canal de control dedicado** se utiliza para la actualización de la ubicación, el registro y el establecimiento de llamadas. En especial, cada BSC mantiene una base de datos de estaciones móviles que se encuentran bajo su jurisdicción, el VLR. La información necesaria para mantener el VLR se envía en el canal de control dedicado.

Por último tenemos el **canal de control común**, el cual se divide en tres subcanales lógicos. El primero de estos subcanales es el **canal de localización**, que la estación base utiliza para anunciar las llamadas entrantes. Cada estación móvil lo monitorea de manera continua en busca de llamadas que debe responder. El segundo es el **canal de acceso aleatorio**, el cual permite a los usuarios solicitar una ranura en el canal de control dedicado. Si dos peticiones colisionan, se distorsionan y hay que volver a intentar más tarde. Mediante el uso de la ranura del canal de control dedicado, la estación puede establecer una llamada. La ranura asignada se anuncia en el tercer subcanal, el **canal de concesión de acceso**.

Por último, la diferencia entre GSM y AMPS está en la forma en que se maneja la entrega. En AMPS, el MSC la administra por completo sin ayuda de los dispositivos móviles. Con las ranuras de tiempo en GSM, el móvil no envía ni recibe la mayor parte del tiempo. Las ranuras inactivas son una oportunidad para que el móvil mida la calidad de la señal y la compare con las otras estaciones base cercanas. Después de hacer esto, envía la información al BSC, que a su vez puede usar esta información para determinar si un móvil sale de una celda y entra a otra, de modo que pueda realizar la entrega. A este diseño se le conoce como **MAHO (Entrega Asistida por Móvil**, del inglés *Mobile Assisted HandOff*).

2.7.3 Teléfonos móviles de tercera generación (3G): voz y datos digitales

La primera generación de teléfonos móviles era de voz analógica y la segunda generación era de voz digital. La tercera generación de teléfonos móviles, o **3G** como se le conoce comúnmente, es de voz *y* datos digitales.

Hay varios factores que impulsan la industria. Primero, el tráfico de datos ya es mayor que el tráfico de voz en la red fija y está creciendo en forma exponencial, mientras que el tráfico de voz es en esencia fijo. Muchos expertos en la industria esperan también que el tráfico de datos domine al tráfico de voz en los dispositivos móviles muy pronto. Segundo, las industrias de telefonía, entretenimiento y computadoras se han vuelto digitales y están convergiendo con rapidez. Muchas personas suspiran por los dispositivos ligeros y portátiles que actúan como teléfono, reproductor de música y de video, terminal de correo electrónico, interfaz web, máquina de juegos y mucho más, todo con conectividad inalámbrica mundial a Internet con un alto ancho de banda.

El iPhone de Apple es un buen ejemplo de este tipo de dispositivo 3G. Con él, las personas se envician en los servicios de datos inalámbricos y, en consecuencia, los volúmenes de datos inalámbricos de AT&T se elevan estrepitosamente junto con la popularidad del iPhone. El problema es que este teléfono utiliza una red 2.5G (una red 2G mejorada, pero no es una verdadera red 3G) y no hay suficiente capacidad de datos para mantener felices a los usuarios. El único fin de telefonía móvil 3G es proveer suficiente ancho de banda inalámbrico para mantener felices a estos futuros usuarios.

La ITU trató de ser un poco más específica sobre esta visión allá por el año de 1992. Emitió un plano detallado para alcanzar este sueño, llamado **IMT-2000**. IMT son las siglas en inglés de **Telecomunicaciones Móviles Internacionales**. Los supuestos servicios básicos que la red IMT-2000 debía proveer a sus usuarios son:

1. Transmisión de voz de alta calidad.
2. Mensajería (para reemplazar al correo electrónico, fax, SMS, chat, etcétera).
3. Multimedia (reproducir música, ver videos, películas, televisión, etcétera).
4. Acceso a Internet (navegar por web, incluyendo las páginas con audio y video).

Algunos servicios adicionales podrían ser la videoconferencia, la telepresencia, los juegos en grupo y el comercio móvil (pasar su teléfono por el cajero para pagar en una tienda). Además, se supone que todos estos servicios estén disponibles a nivel mundial (con conexión automática vía satélite al no encontrar una red terrestre), de manera instantánea (siempre conectado) y con calidad de servicio garantizada.

La ITU visualizó una sola tecnología a nivel mundial para IMT-2000, de modo que los fabricantes pudieran construir un solo dispositivo que se pudiera vender y utilizar en cualquier parte del mundo (al igual que los reproductores de CD y las computadoras, y a diferencia de los teléfonos y televisiones móviles). Al tener una sola tecnología también se facilitaría la vida de los operadores de red y se alentaría a más personas a utilizar los servicios. Las guerras de formato, como la batalla entre Betamax y VHS en las videograbadoras, no son buenas para los negocios.

En resumidas cuentas, era un poco optimista. El número 2000 representaba tres cosas: (1) el año en el que se suponía iba a entrar en servicio; (2) la frecuencia con que se suponía que operaría (en MHz); y (3) el ancho de banda que debería tener el servicio (en kbps). Pero no se cumplió nada de lo anterior. No se implementó nada en el 2000. La ITU recomendó que todos los gobiernos reservaran espectro de 2 GHZ para que los dispositivos pudieran operar sin problemas en cualquier país. China reservó el ancho de banda requerido, pero nadie más lo hizo. Finalmente admitieron que 2 Mbps no son viables para usuarios que se desplazan *demasiado* (debido a la dificultad de llevar a cabo las entregas con la suficiente rapidez). Es más realista tener 2 Mbps para usuarios estacionarios en interiores (lo cual competirá directamente con ADSL), 384 kbps para los peatones y 144 kbps para las conexiones en los autos.

A pesar de estos contratiempos iniciales, se ha logrado mucho desde entonces. Se hicieron varias propuestas del IMT y, después de algunas exclusiones, quedaron dos importantes. La primera, **WCDMA** (**CDMA de banda ancha**, del inglés *Wideband CDMA*), fue propuesta por Ericsson e impulsada por la Unión Europea, que la llamó **UMTS** (**Sistema Universal de Telecomunicaciones Móviles**, del inglés *Universal Mobile Telecommunications System*). El otro contendiente era **CDMA2000**, propuesto por Qualcomm.

Ambos sistemas eran más similares que distintos, en cuanto a que se basaban en el CDMA de banda ancha; WCDMA utiliza canales de 5 MHz y CDMA2000 usa canales de 1.25 MHz. Si los ingenieros de Ericsson y Qualcomm se reunieran en un cuarto y se les pidiera que crearan un diseño común, es probable que pudieran hacerlo con bastante rapidez. El problema real no es la ingeniería, sino la política (como siempre). Europa quería un sistema que interactuara con GSM, mientras que Estados Unidos quería un sistema compatible con uno que ya se distribuía ampliamente en ese país (IS-95). Cada lado también apoyaba a su compañía local (Ericsson tiene su sede en Suecia; Qualcomm está en California). Finalmente, Ericsson y Qualcomm se involucraron en numerosas demandas por sus respectivas patentes de CDMA.

A nivel mundial, entre un 10% y 15% de los suscriptores móviles ya utilizan tecnologías 3G. En Norteamérica y Europa, cerca de una tercera parte de los suscriptores móviles están en 3G. Japón adoptó esta tecnología desde un inicio y ahora casi todos los teléfonos móviles en este país son 3G. Estas cifras incluyen tanto a UMTS como a CDMA2000; 3G sigue siendo una gran caldera en actividad a medida que el mercado se sacude. Para aumentar la confusión, UMTS se convirtió en un solo estándar 3G con varias opciones incompatibles, incluyendo CDMA2000. Este cambio fue un esfuerzo por unificar los diferentes campos, pero sólo oculta las diferencias técnicas y oscurece el enfoque de los esfuerzos continuos. Utilizaremos UMTS para indicar WCDMA y diferenciarlo de CDMA2000.

Enfocaremos nuestro estudio en el uso de CDMA en las redes celulares, puesto que es la característica distintiva de ambos sistemas. CDMA no es FDM ni TDM, sino más bien un tipo de mezcla en la que cada usuario envía sobre la misma banda de frecuencia al mismo tiempo. Cuando se propuso por primera vez para los sistemas celulares, la industria obtuvo casi la misma reacción que obtuvo Colón de la reina Isabel cuando le propuso llegar a India navegando en la dirección contraria. Sin embargo y a través de la persistencia de una sola compañía (Qualcomm), CDMA triunfó como sistema 2G (IS-95) y maduró al punto en que se convirtió en la base técnica para 3G.

Para que CDMA funcione en el ambiente de la telefonía móvil, se requiere algo más que la técnica CDMA básica que describimos en la sección anterior. Específicamente, describimos el CDMA sincrónico, en donde las secuencias de chips son exactamente ortogonales. Este diseño funciona cuando todos los usuarios están sincronizados en el tiempo inicial de sus secuencias de chips, como cuando la estación base transmite a los móviles. La estación base puede transmitir las secuencias de chips iniciando al mismo tiempo, de modo que las señales sean ortogonales y se puedan separar. Sin embargo, es difícil sincronizar las transmisiones de los teléfonos móviles por separado. Si no tenemos cuidado, sus transmisiones llegarían a la estación base en distintos tiempos, sin garantía de ortogonalidad. Para que los móviles puedan enviar a la estación base sin sincronización, necesitamos secuencias de códigos que sean ortogonales entre sí en todos los posibles desplazamientos, no sólo cuando se alinean en un principio.

Aunque no es posible encontrar secuencias que sean exactamente ortogonales para este caso general, las secuencias seudoaleatorias largas se acercan lo suficiente. Poseen la propiedad de que, con una alta probabilidad, tienen una baja **correlación cruzada** entre sí en todos los desplazamientos. Esto significa que, cuando una secuencia se multiplica por otra y se suman para calcular el producto interno, el resultado será pequeño; sería cero si fueran ortogonales (instintivamente, las secuencias aleatorias siempre deberían ser distintas unas de otras. Al multiplicarlas se debería producir una señal aleatoria, lo cual se resumirá en un resultado pequeño). Esto permite a un receptor filtrar las transmisiones no deseadas para sacarlas de la señal recibida. Además, la **autocorrelación** de las secuencias seudoaleatorias también es pequeña, con alta probabilidad, excepto en un desplazamiento de cero. Esto significa que cuando se multiplique una secuencia por una copia retrasada de sí misma y se sume, el resultado será pequeño excepto cuando el retardo sea cero (instintivamente, una secuencia aleatoria retrasada es similar a una secuencia aleatoria diferente, y regresamos al caso de la correlación cruzada). Esto permite a un receptor fijarse en el inicio de la transmisión deseada en la señal recibida.

El uso de secuencias seudoaleatorias permite a la estación base recibir mensajes CDMA de móviles que no estén sincronizados. No obstante, una suposición implícita en nuestro estudio del CDMA es que los niveles de potencia de todos los móviles son iguales en el receptor. Si no lo son, una pequeña correlación cruzada con una señal poderosa podría superar a una autocorrelación grande con una señal débil. En consecuencia, hay que controlar la potencia de transmisión en los móviles para minimizar la interferencia entre señales en disputa. Es esta interferencia la que limita la capacidad de los sistemas CDMA.

Los niveles de potencia recibidos en una estación base dependen de qué tan lejos están los transmisores y de cuánta potencia transmiten. Puede haber muchas estaciones móviles a diferentes distancias de la estación base. Una buena heurística para igualar la potencia recibida es que cada estación móvil transmita a la estación base con el inverso del nivel de potencia que recibe de la estación base. En otras palabras, una estación móvil que reciba una señal débil de la estación base usará más potencia que una estación que reciba una señal fuerte. Para una mayor precisión, la estación base también puede proporcionar retroalimentación a cada móvil para que aumente, disminuya o mantenga estable su potencia de transmisión. La retroalimentación debe ser frecuente (1 500 veces por segundo) puesto que un buen control de la potencia es importante para minimizar la interferencia.

Otra mejora en comparación con el esquema CDMA básico que describimos antes es permitir que distintos usuarios envíen datos a distintas tasas de transmisión. Para lograr este truco en CDMA se fija la tasa a la que se transmiten los chips y se asignan a los usuarios secuencias de chips de distintas longitudes. Por ejemplo, en WCDMA la tasa de transmisión de chips es de 3.84 Mchips/seg y los códigos de dispersión varían de 4 a 256 chips. Con un código de 256 chips, quedan cerca de 12 kbps después de la corrección de errores; esta capacidad es suficiente para una llamada de voz. Con un código de cuatro chips, la tasa de transmisión de datos de usuario se aproxima a 1 Mbps. Los códigos de longitud intermedia proporcionan tasas de transmisión intermedias; para obtener varios Mbps, el móvil debe usar más de un canal de 5 MHz a la vez.

Ahora vamos a describir las ventajas de CDMA, dado que ya hablamos sobre los problemas para hacer que funcione. Tiene tres ventajas principales. Primero, CDMA puede mejorar la capacidad al aprovechar los pequeños periodos cuando algunos transmisores están en silencio. En las llamadas de voz corteses, una persona permanece en silencio mientras la otra habla. En promedio, la línea está ocupada sólo un 40% del tiempo. Sin embargo, las pausas pueden ser pequeñas y difíciles de predecir. En los sistemas TDM o FDM no es posible reasignar ranuras de tiempo o canales de frecuencia con la suficiente rapidez como para poder beneficiarse de estos pequeños silencios. No obstante, en CDMA un usuario puede reducir la interferencia para otros usuarios con el simple hecho de no transmitir, y es probable que una parte de los usuarios no transmitan en una celda ocupada, en un momento dado. Así, CDMA aprovecha los silencios esperados para permitir un mayor número de llamadas simultáneas.

Segundo, en CDMA cada celda usa las mismas frecuencias. A diferencia de GSM y AMPS, no se necesita FDM para separar las transmisiones de los distintos usuarios. Esto elimina las complicadas tareas de planificación de frecuencia y mejora la capacidad. Además, una estación base puede utilizar con facilidad varias antenas direccionales, o **antenas por sectores**, en vez de una antena omnidireccional. Las antenas direccionales concentran una señal en la dirección deseada y reducen la señal (y por ende la interferencia) en otras direcciones. Esto a su vez incrementa la capacidad. Hay tres diseños de sectores comunes. La estación base debe rastrear el móvil a medida que se desplaza de un sector a otro. Este rastreo es muy sencillo en CDMA debido a que se utilizan todas las frecuencias en todos los sectores.

Tercero, CDMA facilita la **entrega suave** (*soft handoff*), en donde el móvil es adquirido por la nueva estación base antes de que la anterior se desconecte. De esta manera no se pierde la continuidad. En la figura 2-49 se muestra la entrega suave. Es fácil de llevar a cabo en CDMA debido a que se utilizan todas las frecuencias en cada celda. La alternativa es una **entrega dura** (*hard handoff*), en donde la estación base anterior se desconecta del móvil antes de que la nueva lo adquiera. Si la nueva estación base no puede adquirirlo (por ejemplo, si no hay una frecuencia disponible), la llamada se termina en forma abrupta. Los usuarios suelen notar esto, pero es inevitable en ocasiones debido al diseño actual. La entrega dura es la norma en los diseños FDM para evitar el costo de que el móvil transmita o reciba en dos frecuencias al mismo tiempo.

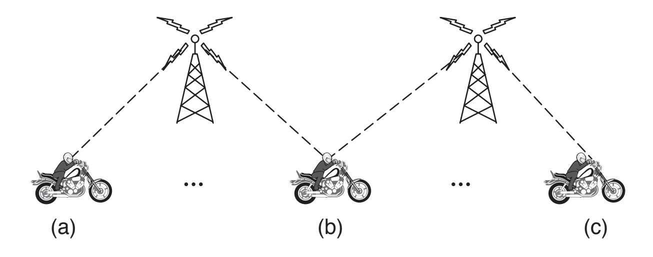


Figura 2-49. Entrega suave (a) antes, (b) durante y (c) después.

Se ha escrito mucho sobre 3G, la mayor parte de la gente lo describe como lo mejor que ha pasado desde que se inventó el pan en rebanadas. Mientras tanto, algunas operadoras han avanzado con cautela en dirección de 3G hacia lo que se conoce algunas veces como **2.5G**, aunque tal vez 2.1G sería más preciso. Uno de esos sistemas es **EDGE** (**Tasa de Datos Mejorada para la Evolución de GSM**, del inglés *Enhanced Data rates for GSM Evolution*), que viene siendo simplemente GSM con más bits por símbolo. El problema es que más bits por símbolo también implican más errores por símbolo, por lo que EDGE tiene nueve distintos esquemas para modulación y corrección de errores, los cuales difieren en la cantidad de ancho de banda que se dedica para corregir los errores introducidos por el aumento en la velocidad. EDGE es un paso a lo largo de una trayectoria evolutiva que se define desde GSM hasta WCDMA. De

manera similar, existe una trayectoria evolutiva para que los operadores se actualicen de las redes IS-95 a CDMA2000.

Aun cuando todavía no se implementan totalmente las redes 3G, algunos investigadores lo dan por sentado. Estas personas ya están trabajando en sistemas 4G bajo el nombre de **LTE** (**Evolución a Largo Plazo**, del inglés *Long Term Evolution*). Algunas de las características propuestas de 4G incluyen: alto ancho de banda, ubicuidad (conectividad en cualquier parte), integración perfecta con otras redes IP alámbricas e inalámbricas (incluyendo los puntos de acceso 802.11), manejo de espectro y recursos adaptable, y una alta calidad de servicio para multimedia.

Mientras tanto, ya hay disponibles redes inalámbricas con niveles 4G de rendimiento. El principal ejemplo es **802.16**, también conocida como **WiMAX**. Para ver las generalidades sobre WiMAX móvil, consulte a Ahmadi (2009). Decir que la industria está en un estado de cambio es muy poco. Eche un vistazo en unos cuantos años para ver lo que ha pasado.

2.8 TELEVISIÓN POR CABLE

Hemos estudiado tanto los sistemas telefónicos fijos como los inalámbricos con suficiente detalle. Ambos jugarán un papel importante en las redes futuras. Sin embargo, hay un participante importante que surgió durante la última década para el acceso a Internet: las redes de televisión por cable. En la actualidad muchas personas obtienen su servicio de teléfono y de Internet a través de cable. En las siguientes secciones analizaremos con más detalle el sistema de televisión por cable como una red y lo compararemos con los sistemas telefónicos que acabamos de estudiar. Algunas referencias relevantes para obtener más información son Donaldson y Jones (2001), Dutta-Roy (2001) y Fellows y Jones (2001).

2.8.1 Televisión por antena comunal

La televisión por cable se concibió a finales de la década de 1940 como una forma de proporcionar mejor recepción a las personas que viven en áreas rurales o montañosas. En un principio, el sistema consistía en una antena grande en la cima de una colina para captar la señal de televisión, un amplificador conocido como **amplificador de cabecera** (*head end amplifier*) para reforzarla y un cable coaxial para enviarla a las casas de las personas, como se ilustra en la figura 2-50.

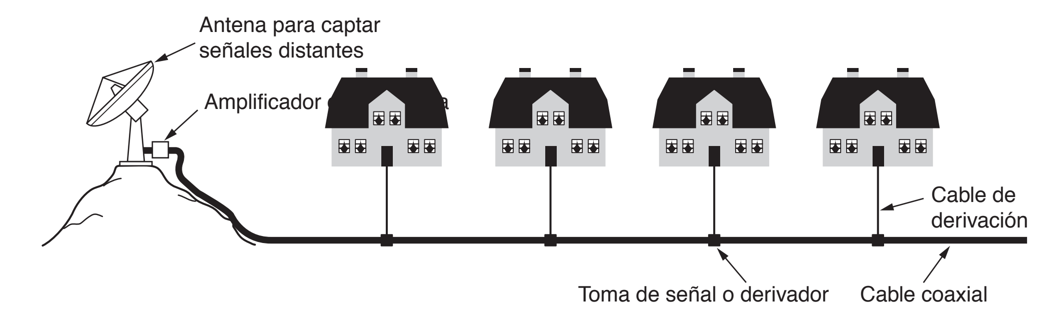


Figura 2-50. Uno de los primeros sistemas de televisión por cable.

Durante los primeros años, a la televisión por cable se le llamaba **Televisión por antena comunal**. Era en esencia un negocio familiar; cualquiera que fuera hábil con la electrónica podía establecer un

servicio para su comunidad, y los usuarios podían pagarlo en conjunto. A medida que creció el número de suscriptores, se empalmaban cables adicionales al cable original y se agregaban amplificadores según fuera necesario. La transmisión era de una vía, del amplificador de cabecera a los usuarios. Para 1970 ya existían miles de sistemas independientes.

En 1974, Time, Inc. inició un nuevo canal llamado Home Box Office, con contenido nuevo (películas) que se distribuía sólo por cable. Le siguieron otros canales que se transmitían sólo por cable enfocados a noticias, deportes, cocina y muchos otros temas más. Este desarrollo dio origen a dos cambios en la industria. Primero, las grandes compañías comenzaron a comprar sistemas de cable existentes y a tender nuevo cable para adquirir más suscriptores. Segundo, surgió la necesidad de conectar múltiples sistemas, por lo general en ciudades distantes, para distribuir los nuevos canales por cable. Las compañías de cable comenzaron a instalar cable entre ciudades para conectarlas en un solo sistema. Este patrón fue similar a lo que pasó en la industria telefónica 80 años antes con la conexión de las oficinas finales locales previamente aisladas para que se pudieran hacer llamadas de larga distancia.

2.8.2 Internet por cable

A través de los años, el sistema de televisión por cable creció y los cables entre las distintas ciudades se reemplazaron por fibra de ancho de banda alto, de manera similar a lo que sucedió con el sistema telefónico. Un sistema en el que se utiliza fibra para distancias considerables y cable coaxial para las casas es conocido como sistema **HFC** (**Red Híbrida de Fibra Óptica y Cable Coaxial**, del inglés *Hybrif Fiber Coax*). Los convertidores electroópticos que interactúan entre las partes óptica y eléctrica del sistema se llaman **nodos de fibra**. Debido a que el ancho de banda de la fibra es mucho mayor al del cable coaxial, un nodo de fibra puede alimentar múltiples cables coaxiales. En la figura 2-51(a) se muestra parte de un sistema HFC moderno.

Durante la década pasada, muchos operadores de cable decidieron entrar al negocio de acceso a Internet y con frecuencia también al de la telefonía. Las diferencias técnicas entre la planta de cable y la de telefonía tuvieron un efecto con respecto a lo que se debía hacer para alcanzar esas metas. Por un lado, hubo que reemplazar todos los amplificadores de una vía del sistema por amplificadores de dos vías para soportar las transmisiones ascendentes y descendentes. Mientras esto ocurría, los primeros sistemas de Internet por cable utilizaban la red de televisión por cable para las transmisiones descendentes y una conexión de marcación por medio de la red telefónica para las transmisiones ascendentes. Era una solución provisional muy astuta, pero no se podía considerar una verdadera red.

Sin embargo, hay otra diferencia entre el sistema HFC de la figura 2-51(a) y el sistema telefónico de la figura 2-51(b) que es más difícil eliminar. En los vecindarios, muchas casas comparten un solo cable, mientras que en el sistema telefónico, cada casa tiene su propio lazo local privado. Cuando se emplea en la difusión de televisión, esta compartición no tiene importancia. Todos los programas se difunden a través del cable y no importa si hay 10 o 10 000 televidentes. Pero cuando el mismo cable se utiliza para el acceso a Internet, el hecho de que haya 10 o 10 000 usuarios tiene mucha importancia. Si un usuario decide descargar un archivo muy grande, ese ancho de banda se les resta a otros usuarios. Entre más usuarios haya, habrá más competencia por el ancho de banda. El sistema telefónico no tiene esta propiedad particular: descargar un archivo grande a través de una línea ADSL no reduce el ancho de banda del vecino. Por otra parte, el ancho de banda del cable coaxial es mucho mayor que el del cable de par trenzado, por lo que usted se podría considerar afortunado si sus vecinos no pasan mucho tiempo en Internet.

La forma en que la industria del cable ha lidiado con este problema es dividir los cables largos y conectar cada uno de ellos directamente a un nodo de fibra. El ancho de banda que el amplificador de cabecera proporciona a cada nodo de fibra es efectivamente infinito, de modo que mientras no haya demasiados suscriptores en cada segmento del cable, la cantidad de tráfico será manejable. En la actualidad,

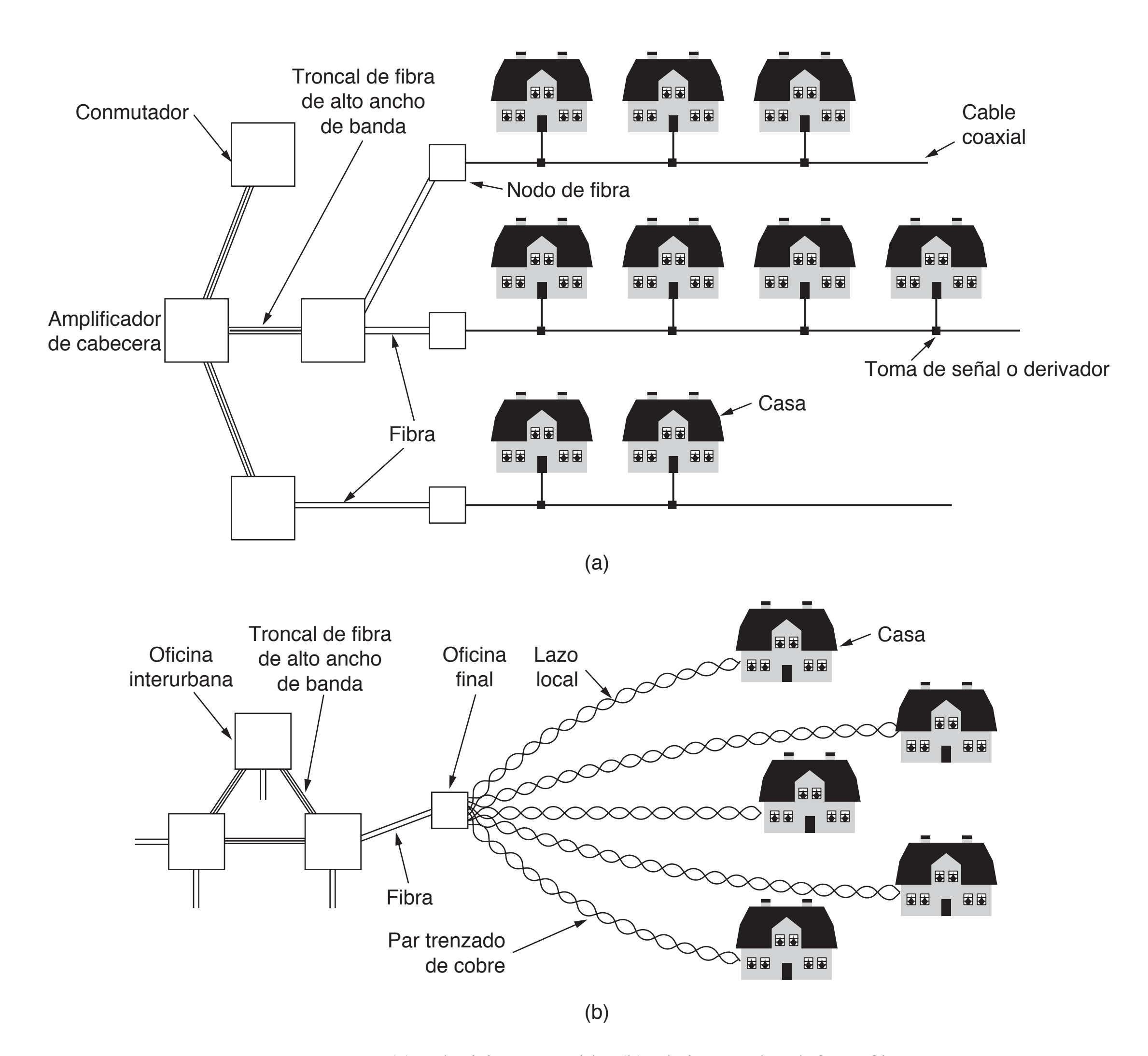


Figura 2-51. (a) Televisión por cable. (b) El sistema de telefonía fija.

los cables típicos tienen de 500 a 2000 casas, pero entre más y más gente se suscribe a Internet por cable, la carga podría volverse demasiada, lo que requeriría más divisiones y más nodos de fibra.

2.8.3 Asignación de espectro

Es probable que al deshacerse de todos los canales de TV y utilizar la infraestructura de cable tan sólo para el acceso a Internet se genere una cantidad considerable de clientes iracundos, razón por la cual las compañías de cable dudan en hacerlo. Además, la mayoría de las ciudades regulan estrictamente lo que hay en el cable, de modo que los operadores de cable tal vez no podrían hacer esto aunque realmente desearan hacerlo. Como consecuencia, necesitan encontrar una manera de que las televisiones e Internet coexistan pacíficamente en el mismo cable.

La solución es usar la multiplexión por división de frecuencia. Los canales de televisión por cable en Norteamérica ocupan la región de 54 a 550 MHz (excepto por la radio FM de 88 a 108 MHz). Estos canales

tienen 6 MHz de ancho, incluyendo las bandas de guarda, y pueden transportar un canal tradicional de televisión analógica o varios canales de televisión digital. En Europa el extremo inferior por lo general es de 65 MHz y los canales tienen un ancho de 6 a 8 MHz para la resolución más alta requerida por PAL y SECAM, pero en lo demás el esquema de asignación es similar. No se utiliza la parte baja de la banda. Los cables modernos también pueden operar muy por encima de 550 MHz, por lo general a 750 MHz o más. La solución elegida fue introducir canales ascendentes en la banda de 5 a 42 MHz (un poco más arriba en Europa) y utilizar las frecuencias en el extremo superior para las señales descendentes. El espectro del cable se ilustra en la figura 2-52.

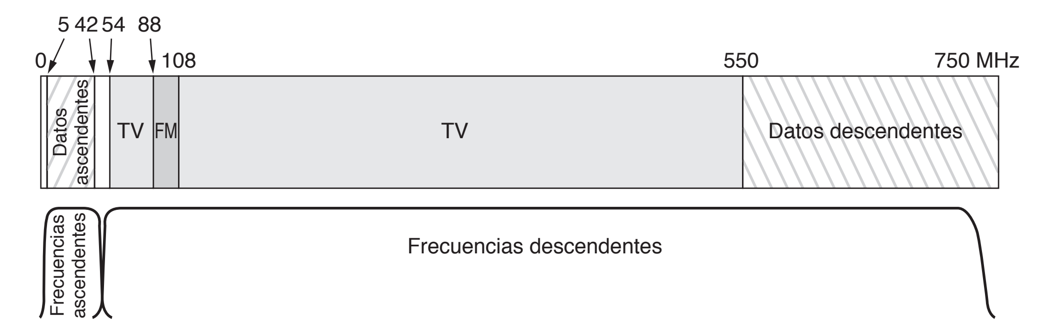


Figura 2-52. Asignación de frecuencia en un sistema típico de TV por cable utilizado para el acceso a Internet.

Hay que tener en cuenta que como todas las señales de televisión son descendentes, es posible utilizar amplificadores ascendentes que funcionen sólo en la región de 5 a 42 MHz y amplificadores descendentes que sólo funcionen a 54 MHz o más, como se muestra en la figura. Así, obtenemos una asimetría en los anchos de banda de los flujos ascendente y descendente debido a que hay más espectro disponible sobre la televisión que debajo de ella. Por otra parte, la mayoría de los usuarios desean más tráfico descendente y los operadores de cable están felices con este hecho de la vida. Como vimos antes, las compañías telefónicas por lo general ofrecen un servicio DSL asimétrico, aunque no tienen ninguna razón técnica para hacerlo.

Además de actualizar los amplificadores, el operador también tiene que actualizar el amplificador de cabecera, de un amplificador tonto a un sistema de cómputo digital inteligente con una interfaz de fibra de ancho de banda alto conectada a un ISP. A menudo el nombre también se actualiza, de “amplificador de cabecera” a **CMTS** (**Sistema de Terminación del Módem de Cable**, del inglés *Cable Modem Termination System*). En el siguiente texto evitaremos la actualización de nombre y nos apegaremos al tradicional término “amplificador de cabecera”.

2.8.4 Módems de cable

Para acceder a Internet se requiere un módem de cable, un dispositivo que tiene dos interfaces: una en la computadora y la otra en la red de cable. Durante los primeros años de Internet por cable, cada operador tenía un módem de cable propietario, que era instalado por un técnico de la compañía de cable. Sin embargo, pronto quedó claro que un estándar abierto podría crear un mercado de módems de cable competitivo y bajar los precios, con lo que se alentaría el uso del servicio. Además, al permitir que los clientes compren los módems de cable en tiendas y que los instalen ellos mismos (al igual que los puntos de acceso inalámbricos) se podrían eliminar las temidas cuadrillas de la compañía de cable.

En consecuencia, los operadores de cable más grandes se unieron a una compañía llamada CableLabs para producir un estándar de módem de cable y probar la compatibilidad de los productos. Este estándar, llamado **DOCSIS** (**Especificación de Interfaz para Servicio de Datos por Cable**, del inglés *Data Over Cable Service Interface Specification*), ha reemplazado casi a todos los módems propietarios. La versión 1.0 de DOCSIS salió en 1997 y pronto le siguió DOCSIS 2.0 en 2001. Aumentó las tasas de transmisión ascendente para ofrecer un mejor soporte a los servicios simétricos tales como la telefonía IP. La versión más reciente del estándar es DOCSIS 3.0, que salió en 2006. Usa más ancho de banda para incrementar las tasas de transmisión en ambos sentidos. La versión europea de estos estándares se llama **EuroDOCSIS**. Sin embargo, no a todos los operadores de cable les gusta la idea de un estándar, debido a que muchos de ellos estaban ganando bastante dinero rentando sus módems a sus clientes cautivos. Un estándar abierto con docenas de fabricantes vendiendo módems de cable en tiendas termina con esta práctica tan lucrativa.

La interfaz módem a computadora es directa. Por lo general es Ethernet y en ocasiones es USB. El otro extremo es más complicado, ya que usa FDM, TDM y CDMA para compartir el ancho de banda del cable entre los suscriptores.

Cuando un módem de cable se conecta y enciende, explora los canales descendentes en busca de un paquete especial que el amplificador de cabecera transmite periódicamente para proporcionar parámetros del sistema a los módems que se acaban de conectar. Al encontrar este paquete, el nuevo módem anuncia su presencia en uno de los canales ascendentes. El amplificador de cabecera responde y asigna el módem a sus canales ascendente y descendente. Estas asignaciones se pueden cambiar después, si el amplificador de cabecera considera necesario balancear la carga.

El uso de canales de 6 u 8 MHz es la parte correspondiente a FDM. Cada módem de cable envía datos en un canal ascendente y en un canal descendente, o en varios canales si se usa el estándar DOCSIS 3.0. El esquema usual es tomar cada canal descendente de 6 (u 8) MHz y modularlo con QAM-64 o, si la calidad del cable es excepcionalmente buena, con QAM-256. Con un canal de 6 MHz y QAM-64 obtenemos casi 36 Mbps. Cuando se resta la sobrecarga, la carga útil neta es de cerca de 27 Mbps. Con QAM-256 la carga útil neta aproximada es de 39 Mbps. Los valores europeos son 1/3 más grandes.

En el flujo ascendente hay más ruido de RF puesto que el sistema no estaba diseñado originalmente para datos, y el ruido de los múltiples suscriptores se canalizan hacia el amplificador de cabecera, por lo que se utiliza un esquema más conservador que varía de QPSK a QAM-128, en donde algunos de los símbolos se utilizan para protección contra errores mediante la modulación codificada de Trellis. Con menos bits por símbolo en el flujo ascendente, la asimetría entre las tasas de transmisión ascendente y descendente es mucho más de lo que sugiere la figura 2-52.

Después se utiliza TDM para compartir el ancho de banda en el flujo ascendente entre los múltiples suscriptores. Si no fuera así, sus transmisiones colisionarían en el amplificador de cabecera. El tiempo se divide en **minirranuras** y los distintos suscriptores envían en distintas minirranuras. Para que esto funcione, el módem determina su distancia desde el amplificador de cabecera enviándole un paquete especial y espera a ver cuánto tiempo tarda en llegar la respuesta. Este proceso se conoce como **alineación** (*ranging*). Es importante que el módem conozca su distancia para que esté bien sincronizado. Cada paquete ascendente se debe ajustar en una o más minirranuras consecutivas en el amplificador de cabecera al momento de recibirlo. El amplificador de cabecera anuncia el inicio de una nueva ronda de minirranuras en forma periódica, pero la señal de partida no se escucha en todos los módems al mismo tiempo debido al tiempo de propagación en el cable. Al saber qué tan lejos se encuentra del amplificador de cabecera, cada módem puede calcular hace cuánto tiempo empezó realmente la primera minirranura. La longitud de las minirranuras es dependiente de la red. Una carga útil típica es de 8 bytes.

Durante la inicialización, el amplificador de cabecera asigna a cada módem una minirranura con el fin de utilizarla para solicitar el ancho de banda ascendente. Cuando una computadora desea enviar un paquete, transfiere ese paquete al módem, que a su vez solicita el número necesario de minirranuras para

el paquete. Si se acepta la solicitud, la cabecera coloca una confirmación en el canal descendente para indicar al módem cuáles minirranuras se reservaron para su paquete. Después se envía el paquete, empezando en la minirranura asignada para este fin. Los paquetes adicionales se pueden solicitar mediante el uso de un campo en el encabezado.

Como regla, la misma minirranura se asignará a múltiples módems, lo que generará una contienda. Existen dos diferentes posibilidades para lidiar con ello. La primera es que se utilice CDMA para compartir la minirranura entre los suscriptores. Esto resuelve el problema de la contienda, debido a que todos los suscriptores con una secuencia de código CDMA pueden enviar al mismo tiempo, aunque a una tasa de transmisión reducida. La segunda opción es no utilizar CDMA, en cuyo caso tal vez no se confirme la solicitud debido a una colisión. En este caso, el módem simplemente espera un tiempo aleatorio e intenta de nuevo. Después de cada fracaso sucesivo, se duplica el tiempo aleatorio (para los lectores que ya están familiarizados con las redes, este algoritmo es simplemente ALOHA ranurado con retroceso exponencial binario. No es posible utilizar Ethernet en el cable porque las estaciones no pueden detectar el medio. En el capítulo 4 retomaremos este tema).

Los canales descendentes se manejan de manera distinta a los ascendentes. Para empezar, sólo hay un emisor (el amplificador de cabecera) por lo que no hay contienda ni necesidad de minirranuras, lo que en realidad es tan sólo multiplexión estadística por división de tiempo. Por otro lado, el tráfico descendente por lo general es mucho mayor que el ascendente, de modo que se utiliza un tamaño fijo de paquete de 204 bytes. Parte de esto es un código de corrección de errores Reed-Solomon y cierta sobrecarga, lo que deja una carga útil de usuario de 184 bytes. Estos números se eligieron por compatibilidad con la televisión digital que utiliza MPEG-2, así que los canales descendentes de datos y de TV se formatean de la misma manera. Lógicamente, las conexiones son como se muestra en la figura 2-53.

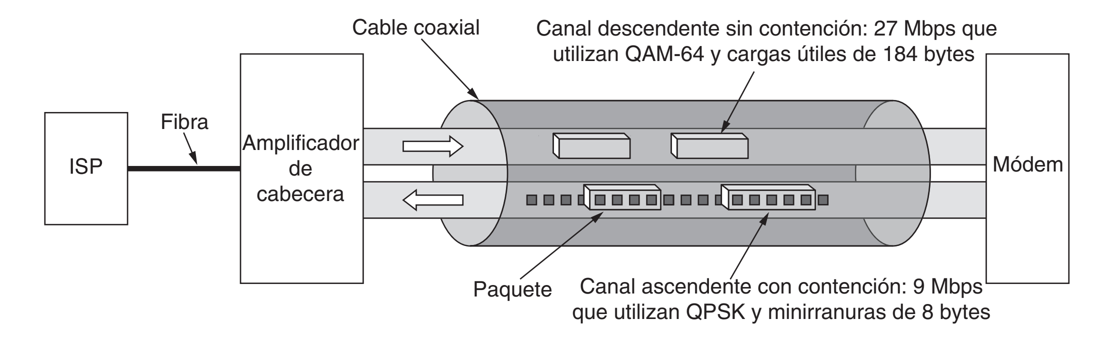


Figura 2-53. Detalles típicos de los canales ascendente y descendente en Norteamérica.

2.8.5 Comparación de ADSL y cable

¿Qué es mejor, ADSL o el cable? Esto es como preguntar qué sistema operativo es mejor. O qué lenguaje es mejor. O qué religión. La respuesta que obtenga depende de a quién le pregunte. Comparemos ADSL y el cable con base en unos cuantos puntos. Los dos utilizan la fibra óptica en la red troncal, pero difieren en el extremo. El cable utiliza cable coaxial; ADSL, cable de par trenzado. La capacidad de carga teórica del cable coaxial es de cientos de veces más que el cable de par trenzado. Sin embargo, la capacidad máxima del cable no está disponible para los usuarios de datos porque la mayor parte del ancho de banda del cable se desperdicia en cosas inútiles, como en programas de televisión.

En la práctica, es difícil generalizar acerca de la capacidad efectiva. Los proveedores de ADSL indican específicamente el ancho de banda (por ejemplo, flujo descendente de 1 Mbps, flujo ascendente

de 256 kbps) y por lo general alcanzan cerca de 80% de esta capacidad de manera consistente. Los proveedores de cable pueden encubrir el ancho de banda de cada usuario para ayudarle a hacer predicciones de rendimiento, pero en realidad no pueden dar garantías pues la capacidad efectiva depende de cuántas personas estén actualmente activas en el segmento de cable del usuario. Algunas veces puede ser mejor que ADSL y otras podría ser peor. Sin embargo, lo que sí puede ser molesto es la incertidumbre. Tener servicio excelente por un minuto no garantiza que al siguiente minuto también lo tendrá, debido a que el ancho de banda más grande de la ciudad puede haber sido acaparado por otra computadora que se acaba de encender.

A medida que un sistema ADSL adquiere más usuarios, este incremento tiene muy poco efecto en los usuarios existentes, debido a que cada uno tiene una conexión dedicada. Con el cable, conforme más personas se suscriban al servicio de Internet, el rendimiento de los usuarios existentes disminuirá. El único remedio es que el operador de cable divida los cables ocupados y conecte en forma directa cada uno a un nodo de fibra óptica. Esto cuesta tiempo y dinero, y son presiones de negocios que se deben evitar.

Además, ya estudiamos otro sistema con un canal compartido como el cable: el sistema telefónico móvil. Aquí un grupo de usuarios (los podríamos llamar compañeros de celda) también comparte una cantidad fija de ancho de banda. Para el tráfico de voz, que es bastante uniforme, el ancho de banda se divide estrictamente en porciones fijas entre los usuarios activos mediante el uso de FDM y TDM. Pero para el tráfico de datos, esta división rígida es muy ineficiente puesto que los usuarios de datos por lo general están inactivos, en cuyo caso se desperdicia el ancho de banda reservado. Al igual que con el cable, se utiliza un medio más dinámico para asignar el ancho de banda compartido.

La disponibilidad es un tema en el que ADSL y el cable difieren. Todas las personas tienen teléfono, pero no todos los usuarios están lo suficientemente cerca de sus oficinas finales para obtener ADSL. Por otro lado, no todos los usuarios tienen cable, pero si usted tiene cable y la compañía proporciona acceso a Internet, puede obtenerlo. Para el nodo de fibra o el amplificador de cabecera, la distancia no es un problema. También vale la pena mencionar que debido a que el cable inició como un medio de distribución de televisión, pocos negocios cuentan con él.

Puesto que ADSL es un medio de punto a punto, es por naturaleza más seguro que el cable. Cualquier usuario de cable puede leer fácilmente todos los paquetes que pasen por el cable. Por esta razón, cualquier proveedor de cable que se precie de serlo cifrará todo el tráfico en ambas direcciones. Sin embargo, el hecho de que su vecino pueda obtener sus mensajes cifrados es aún menos seguro que el hecho de que no obtenga nada.

Por lo común el sistema telefónico es más confiable que el cable. Por ejemplo, tiene energía de respaldo y continúa trabajando de manera normal incluso durante una falla en la energía. Con el cable, si falla la energía de cualquier amplificador de la cadena, todos los usuarios descendentes experimentarán un corte de manera instantánea.

Por último, la mayoría de los proveedores ADSL ofrece una opción de ISP. Algunas veces la ley los obliga a hacerlo. Éste no siempre es el caso con los operadores de cable.

La conclusión es que ADSL y el cable son tan parecidos como diferentes. Ofrecen servicios comparables y, conforme la competencia entre ellos se avive más, probablemente también ofrezcan precios comparables.

2.9 RESUMEN

La capa física es la base de todas las redes. La naturaleza impone en todos los canales dos límites fundamentales que determinan su ancho de banda. Estos límites son: el límite de Nyquist, que tiene que ver con los canales sin ruido; y el límite de Shannon, para canales con ruido.

Los medios de transmisión pueden ser guiados y no guiados. Los principales medios guiados son el cable de par trenzado, el cable coaxial y la fibra óptica. Los medios no guiados incluyen la radio terrestre, las microondas, el infrarrojo, los láser a través del aire y los satélites.

Los métodos de modulación digital envían bits a través de los medios guiados y no guiados en forma de señales analógicas. Los códigos de línea operan en banda base y las señales se pueden colocar en una banda de paso mediante la modulación de la amplitud, frecuencia y fase de una portadora. Se pueden compartir canales entre los usuarios mediante la multiplexión por división de tiempo, frecuencia y código.

El sistema telefónico es un elemento clave en la mayoría de las redes de área amplia. Sus componentes principales son: lazos locales, troncales y conmutadores. ADSL ofrece velocidades de hasta 40 Mbps sobre el lazo local al dividirlo en muchas subportadoras que operan en paralelo. Esto excede por mucho las tasas de transmisión de los módems telefónicos. Las PON llevan la fibra hasta el hogar para obtener tasas de acceso aún mayores que ADSL.

Las troncales transportan información digital. Se multiplexan con WDM para proveer muchos enlaces de alta capacidad a través de fibras individuales, así como con TDM para compartir cada enlace de tasa de transmisión alta entre los usuarios. Tanto la conmutación de circuitos como la conmutación de paquetes son importantes.

Para las aplicaciones móviles, el sistema telefónico fijo no es adecuado. En la actualidad los teléfonos móviles se están usando ampliamente para voz y cada vez más para datos. Han pasado por tres generaciones. La primera generación, 1G, fue analógica y estaba bajo el dominio de AMPS. La 2G fue digital, con GSM actualmente el sistema de telefonía móvil más implementado en el mundo. La 3G es digital y se basa en la tecnología CDMA de banda ancha, aunque también se están implementando WCDMA y CDMA2000.

El sistema de televisión por cable es un sistema alternativo para acceso a red. Evolucionó de manera gradual del cable coaxial a una red híbrida de fibra óptica y cable coaxial, y de la televisión a televisión e Internet. Potencialmente, ofrece un ancho de banda muy alto, pero en la práctica el ancho de banda real disponible depende mucho de lo que estén haciendo los demás usuarios, ya que es compartido.

PROBLEMAS

1. Calcule los coeficientes de Fourier para la función $f(t) = t$ $(0 \leq t \leq 1)$.
2. Un canal sin ruido de 4 kHz se muestrea cada 1 mseg. ¿Cuál es la tasa de datos máxima? ¿Cómo cambia la tasa de datos máxima si el canal es ruidoso, con una relación señal a ruido de 30 dB?
3. Los canales de televisión tienen un ancho de 6 MHz. ¿Cuántos bits/seg se pueden enviar si se usan señales digitales de cuatro niveles? Suponga que el canal no tiene ruido.
4. Si se envía una señal binaria por un canal de 3 kHz cuya relación señal a ruido es de 20 dB, ¿cuál es la tasa de datos máxima que se puede obtener?
5. ¿Qué relación señal a ruido se necesita para poner una portadora T1 en una línea de 50 kHz?
6. ¿Cuáles son las ventajas de la fibra óptica con respecto al cobre como medio de transmisión? ¿Hay alguna desventaja al usar fibra óptica en vez de cobre?
7. ¿Cuánto ancho de banda existe en 0.1 micras de espectro a una longitud de onda de 1 micra?
8. Se desea enviar una secuencia de imágenes de pantalla de computadora por una fibra óptica. La pantalla es de $2\,560 \times 1\,600$ píxeles y cada píxel ocupa 24 bits. Hay 60 imágenes de pantalla por segundo. ¿Cuánto ancho de banda se necesita y cuántas micras de longitud de onda se necesitan para esta banda a 1.30 micras?
9. ¿Se cumple el teorema de Nyquist para la fibra óptica de monomodo y alta calidad o solamente para el alambre de cobre?

10. A menudo las antenas de radio funcionan mejor cuando el diámetro de la antena es igual a la longitud de la onda de radio. Las antenas prácticas varían desde 1 cm hasta 5 m de diámetro. ¿Qué rango de frecuencias cubre esto?

11. Un rayo láser de 1 mm de diámetro se apunta a un detector de 1 mm de diámetro a 100 m en el techo de un edificio. ¿Cuánta desviación angular (en grados) deberá tener el láser antes de que pierda al detector?

12. Los 66 satélites de órbita baja en el proyecto Iridium se dividen en seis collares alrededor de la Tierra. A la altitud que están utilizando, el periodo es de 90 minutos. ¿Cuál es el intervalo promedio de entregas de celdas para un transmisor fijo?

13. Calcule el tiempo de tránsito de extremo a extremo para un paquete en los satélites GEO (altitud: 35 800 km), MEO (altitud: 18 000 km) y LEO (altitud: 750 km).

14. Cuál es la latencia de una llamada originada en el Polo Norte para llegar al Polo Sur, si la llamada se encamina mediante satélites Iridium? Suponga que el tiempo de conmutación en los satélites es de 10 microsegundos y que el radio de la Tierra es de 6 371 km.

15. ¿Cuál es el mínimo ancho de banda necesario para alcanzar una tasa de datos de *B* bits/seg si la señal se transmite mediante codificación NRZ, MLT-3 y Manchester? Explique su respuesta.

16. Demuestre que en la codificación 4B/5B ocurrirá una transición de señal por lo menos cada cuatro tiempos de bit.

17. ¿Cuántos códigos de oficina final había antes de 1984, cuando cada oficina final tenía el nombre de los tres dígitos de su código de área y los primeros tres dígitos del número local? Los códigos de área iniciaban con un dígito en el rango de 2 a 9, tenían un 0 o un 1 como su segundo dígito, y terminaban con cualquier dígito. Los primeros dos dígitos de un número local siempre estaban en el rango de 2 a 9. El tercer dígito podía ser cualquiera.

18. Un sistema telefónico simple consiste en dos oficinas finales y una oficina interurbana a la que está conectada cada oficina final mediante una troncal full-dúplex de 1 MHz. En promedio, cada teléfono se usa para hacer cuatro llamadas por cada jornada de 8 horas. La duración media de las llamadas es de 6 minutos. El 10% de las llamadas son de larga distancia (esto es, pasan por la oficina interurbana). ¿Cuál es la cantidad máxima de teléfonos que puede manejar una oficina final? (Suponga que hay 4 kHz por circuito). Explique por qué una compañía telefónica podría decidir soportar un menor número de teléfonos que este número máximo en la oficina final.

19. Una compañía de teléfonos regional tiene 10 millones de suscriptores. Cada uno de sus teléfonos está conectado a una oficina central mediante un cable de par trenzado de cobre. La longitud promedio de estos cables de par trenzado es de 10 km. ¿Cuánto vale el cobre de los lazos locales? Suponga que la sección transversal de cada filamento es un círculo de 1 mm de diámetro, que la densidad del cobre es de 9.0 gramos/cm^3 y que el cobre se vende a $6 por kilogramo.

20. ¿Es un gasoducto un sistema simplex, semi-dúplex, full-dúplex, o ninguno de los anteriores?

21. El costo de un microprocesador potente se ha reducido a tal grado que ahora es posible incluir uno en cada módem. ¿Cómo afecta esto en el manejo de errores en las líneas telefónicas? Acaso niega la necesidad de comprobación/corrección de errores en la capa 2?

22. Un diagrama de constelación de módem, similar al de la figura 2-23, tiene puntos de datos en las siguientes coordenadas: $(1, 1)$, $(1, -1)$, $(-1, 1)$ y $(-1, -1)$. ¿Cuántos bps puede lograr un módem a 1 200 símbolos/seg con estos parámetros?

23. ¿Cuál es la máxima tasa de bits alcanzable en un módem con el estándar V.32, si la tasa de baudios es de 1 200 y no se utiliza ningún tipo de corrección de errores?

24. ¿Cuántas frecuencias utiliza un módem QAM-64 full-dúplex?

25. Diez señales, cada una de las cuales requiere 4 000 Hz, se multiplexan en un solo canal mediante FDM. ¿Cuál es el ancho de banda mínimo requerido para el canal multiplexado? Suponga que las bandas de guarda tienen un ancho de 400 Hz.

26. ¿Por qué se fijó el tiempo de muestreo de PCM en 125 μseg?

27. ¿Cuál es el porcentaje de sobrecarga en una portadora T1? Es decir, ¿qué porcentaje de los 1.544 Mbps no se entrega al usuario final? ¿Cómo se relaciona con el porcentaje de sobrecarga en las líneas OC-1 u OC-768?

28. Compare la tasa de datos máxima de un canal sin ruido de 4 kHz que utiliza:
(a) Codificación analógica (por ejemplo, QPSK) con 2 bits por muestra.
(b) El sistema T1 de PCM.

29. Si un sistema de portadora T1 pierde la pista dónde está, trata de resincronizarse mediante el uso del primer bit de cada trama. ¿Cuántas tramas se tendrían que inspeccionar en promedio para resincronizarse con una probabilidad de 0.001 de estar en un error?

30. ¿Cuál es la diferencia, si la hay, entre la parte demoduladora de un módem y la parte codificadora de un codec? (Después de todo, ambos convierten señales analógicas a digitales).

31. Los relojes de SONET tienen una tasa de desviación de casi 1 parte en 10^9. ¿Cuánto tiempo tomará para que la desviación iguale el ancho de 1 bit? ¿Ve usted alguna implicación práctica de este cálculo? ¿Qué pasaría si la hubiera?

32. ¿Cuánto tiempo se requerirá para transmitir un archivo de 1 GB de un VSAT a otro mediante el uso de un hub, como se muestra en la figura 2-17? Suponga que el enlace ascendente es de 1 Mbps, el enlace descendente es de 7 Mbps y se utiliza conmutación de circuitos con un tiempo de establecimiento del circuito de 1.2 segundos.

33. Calcule el tiempo de tránsito en el problema anterior si esta vez se utiliza conmutación de paquetes. Suponga que el tamaño del paquete es de 64 KB, el retardo de conmutación en el satélite y en el hub es de 10 microsegundos, y el tamaño del encabezado del paquete es de 32 bytes.

34. En la figura 2-40, la tasa de datos de usuario para OC-3 se estableció en 148 608 Mbps. Muestre cómo se puede derivar este número de los parámetros de OC-3 de SONET. ¿Cuáles serán las tasas de transmisión bruta, SPE, y de datos de usuario de una línea OC-3072?

35. Para acomodar tasas de datos menores que STS-1, SONET tiene un sistema de tributarias virtuales (VT). Una VT es una carga útil parcial que se puede insertar en una trama STS-1 y combinar con otras cargas útiles parciales para llenar la trama de datos. VT1.5 utiliza 3 columnas, VT2 utiliza 4, VT3 utiliza 6 y VT6 utiliza 12 columnas de una trama STS-1. ¿Cuál VT puede acomodar:
(a) un servicio DS-1 (1.544 Mbps)?
(b) un servicio europeo CEPT-1 (2.048 Mbps o E1)?
(c) un servicio DS-2 (6.312 Mbps)?

36. ¿Cuál es el ancho de banda disponible para el usuario en una conexión OC-12c?

37. Tres redes de conmutación de paquetes contienen n nodos cada una. La primera red tiene una topología de estrella con un conmutador central, la segunda es un anillo (bidireccional) y la tercera está interconectada por completo, con una conexión de cada nodo hacia cada uno de los otros nodos. ¿Cuáles son las rutas de transmisión óptima, media y de peor caso en saltos?

38. Compare el retardo al enviar un mensaje de x bits por una trayectoria de k saltos en una red de conmutación de circuitos y en una red de conmutación de paquetes (con carga ligera). El tiempo de establecimiento de circuito es de s segundos, el retardo de propagación es de d segundos por salto, el tamaño del paquete es de p bits y la tasa de datos es de b bps. ¿En qué condiciones tiene un retardo menor la red de paquetes? Explique además las condiciones bajo las que es preferible una red de conmutación de paquetes a una red de conmutación de circuitos.

39. Suponga que se van a transmitir x bits de datos de usuario por una trayectoria de k saltos en una red de conmutación de paquetes como una serie de paquetes, cada uno contiene p bits de datos y h bits de encabezado, donde $x \gg p + h$. La tasa de bits de las líneas es de b bps y el retardo de propagación es nulo. ¿Qué valor de p minimiza el retardo total?

40. En un sistema de telefónico móvil típico con celdas hexagonales se prohíbe reutilizar una banda de frecuencia en una celda adyacente. Si están disponibles 840 frecuencias, ¿cuántas se pueden utilizar en una celda dada?

41. El diseño real de las celdas rara vez es tan regular como se muestra en la figura 2-45. Incluso las formas de las celdas individuales por lo general son irregulares. Dé una posible razón de por qué sucedería esto. ¿Cómo afectan estas formas irregulares a la asignación de frecuencias para cada celda?

42. Realice una estimación aproximada de la cantidad de microceldas PCS con un diámetro de 100 m que se requerirían para cubrir San Francisco (120 km^2).

43. Algunas veces cuando un usuario móvil cruza el límite de una celda a otra, la llamada actual se termina de manera repentina, aunque todos los transmisores y receptores estén funcionando correctamente. ¿Por qué?

44. Suponga que *A*, *B* y *C* transmiten de manera simultánea bits 0 mediante un sistema CDMA con las secuencias de chips que se muestran en la figura 2-28(a). ¿Cuál es la secuencia de chips resultante?

45. Considere una manera diferente de ver la propiedad de ortogonalidad de las secuencias de chips CDMA. Cada bit en un par de secuencias puede o no coincidir. Exprese la propiedad de ortogonalidad en términos de coincidencias y falta de coincidencias.

46. Un receptor CDMA obtiene los siguientes chips: $(-1\ +1\ -3\ +1\ -1\ -3\ +1\ +1)$. Suponiendo las secuencias de chips definidas en la figura 2-28(a), ¿cuáles estaciones transmitieron y qué bits envió cada una?

47. En la figura 2-28 hay cuatro estaciones que pueden transmitir. Suponga que se agregan cuatro estaciones más. Proporcione las secuencias de chips de estas estaciones.

48. En su parte más baja, el sistema telefónico tiene forma de estrella, y todos los lazos locales de un vecindario convergen en una oficina final. En contraste, la televisión por cable consiste en un solo cable largo que pasa por todas las casas del mismo vecindario. Suponga que un futuro cable de TV fuera de fibra óptica de 10 Gbps en lugar de cable de cobre. ¿Podría utilizarse para simular un modelo telefónico en el que todo el mundo tuviera su propia línea privada a la oficina final? Si esto fuera posible, ¿cuántas casas con un teléfono podrían conectarse a una sola fibra óptica?

49. Una compañía de cable decide proporcionar acceso a Internet a través de cable en un vecindario compuesto por 5 000 casas. La compañía utiliza cable coaxial y asignación de espectro que permite un ancho de banda descendente de 100 Mbps por cable. Para atraer clientes la compañía decide garantizar un ancho de banda descendente de por lo menos 2 Mbps a cada casa en cualquier momento. Describa lo que necesita hacer la compañía de cable para ofrecer esta garantía.

50. Tomando en cuenta la asignación de espectro mostrada en la figura 2-52 y la información dada en el texto, ¿cuántos Mbps necesita asignar el sistema por cable al flujo ascendente y cuántos al flujo descendente?

51. ¿Qué tan rápido puede un usuario de cable recibir datos si la red está inactiva? Suponga que la interfaz de usuario es:
(a) Ethernet de 10 Mbps
(b) Ethernet de 100 Mbps
(c) Inalámbrica de 54 Mbps.

52. Multiplexar flujos de datos múltiples STS-1, llamados tributarias, es una función imprescindible en SONET. Un multiplexor 3:1 multiplexa tres tributarias STS-1 de entrada en un flujo STS-3 de salida. Esta multiplexión se realiza byte por byte; es decir, los tres primeros bytes de salida son los primeros bytes de las tributarias 1, 2 y 3, respectivamente. Los siguientes tres bytes de salida son los segundos bytes de las tributarias 1, 2 y 3, respectivamente, etcétera. Escriba un programa que simule este multiplexor 3:1. El programa deberá consistir de cinco procesos. El proceso principal crea cuatro procesos, uno para cada una de las tres tributarias STS-1 y uno para el multiplexor. Cada proceso tributario lee una trama STS-1 de un archivo de entrada como una secuencia de 810 bytes. Tales procesos tributarios envían sus tramas (byte por byte) al proceso multiplexor. Éste recibe los bytes, envía una trama STS-3 (byte por byte) y lo escribe en una salida estándar. Utilice tuberías para la comunicación entre procesos.

53. Escriba un programa para implementar CDMA. Suponga que la longitud de una secuencia de chips es de ocho y que el número de estaciones que transmiten es de cuatro. Su programa debe consistir en tres conjuntos de procesos: cuatro procesos transmisores (t0, t1, t2 y t3), un proceso de unión y cuatro procesos receptores (r0, r1, r2 y r3). El programa principal, que también actúa como el procedo de unión, lee primero cuatro secuencias de

chips (notación bipolar) de la entrada estándar y una secuencia de 4 bits (1 bit por cada proceso transmisor que se va a transmitir), y bifurca cuatro pares de procesos transmisor y receptor. A cada par de procesos transmisor/ receptor (t0, r0; t1, r1; t2, r2; t3, r3) se le asigna una secuencia de chips y a cada proceso transmisor se le asigna un bit 1 (el primer bit a t0, el segundo a t1, etcétera). A continuación, cada proceso transmisor calcula la señal a transmitir (una secuencia de 8 bits) y la envía al proceso de unión. Después de recibir señales de los cuatro procesos transmisores, el proceso de unión combina las señales y envía la señal combinada a los cuatro procesos receptores. A su vez, cada proceso receptor calcula el bit que recibió y lo imprime en la salida estándar. Use tuberías para la comunicación entre procesos.

3

LA CAPA DE ENLACE DE DATOS

En este capítulo estudiaremos los principios de diseño de la segunda capa, la capa de enlace de datos. Este estudio se enfoca en los algoritmos para lograr una comunicación confiable y eficiente de unidades completas de información llamadas **tramas** (en vez de bits individuales, como en la capa física) entre dos máquinas adyacentes. Por adyacente, queremos decir que las dos máquinas están conectadas mediante un canal de comunicaciones que actúa de manera conceptual como un alambre (por ejemplo, un cable coaxial, una línea telefónica o un canal inalámbrico). La propiedad esencial de un canal que lo hace asemejarse a un "alambre" es que los bits se entregan exactamente en el mismo orden en que se enviaron.

A primera vista se podría pensar que este problema es tan trivial que no hay nada que estudiar: la máquina *A* sólo pone los bits en el alambre y la máquina *B* simplemente los toma. Por desgracia, en ocaciones los canales de comunicación cometen errores. Además, sólo tienen una tasa de transmisión de datos finita y hay un retardo de propagación distinto de cero entre el momento en que se envía un bit y el momento en que se recibe. Estas limitaciones tienen implicaciones importantes para la eficiencia de la transferencia de datos. Los protocolos usados para comunicaciones deben considerar todos estos factores. Dichos protocolos son el tema de este capítulo.

Después de una introducción a los aspectos clave de diseño presentes en la capa de enlace de datos, comenzaremos nuestro estudio de sus protocolos mediante un análisis de la naturaleza de los errores y la manera en que se pueden detectar y corregir. Después estudiaremos una serie de protocolos de complejidad creciente, cada uno de los cuales resuelve una cantidad mayor de los problemas presentes en esta capa. Por último, concluiremos con algunos ejemplos de protocolos de enlace de datos.

3.1 CUESTIONES DE DISEÑO DE LA CAPA DE ENLACE DE DATOS

La capa de enlace de datos utiliza los servicios de la capa física para enviar y recibir bits a través de los canales de comunicación. Tiene varias funciones específicas, entre las que se incluyen:

1. Proporcionar a la capa de red una interfaz de servicio bien definida.
2. Manejar los errores de transmisión.
3. Regular el flujo de datos para que los emisores rápidos no saturen a los receptores lentos.

Para cumplir con estas metas, la capa de enlace de datos toma los paquetes que obtiene de la capa de red y los encapsula en **tramas** para transmitirlos. Cada trama contiene un encabezado, un campo de carga útil (*payload*) para almacenar el paquete y un terminador, como se muestra en la figura 3-1. El manejo de las tramas es la tarea más importante de la capa de enlace de datos. En las siguientes secciones examinaremos en detalle todos los aspectos antes mencionados.

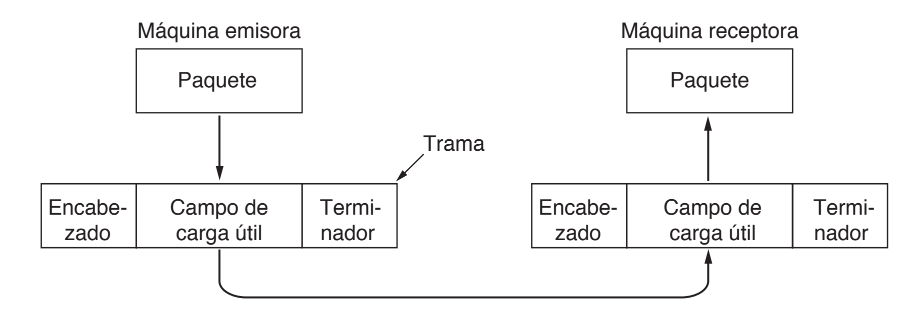


Figura 3-1. Relación entre paquetes y tramas.

Aunque este capítulo trata de manera explícita sobre la capa de enlace de datos y sus protocolos, muchos de los principios que estudiaremos aquí, como el control de errores y el control de flujo, están presentes también en la capa de transporte y en otros protocolos. Esto se debe a que la confiabilidad es una meta general que se logra cuando todas las capas trabajan en conjunto. De hecho, en muchas redes estas funciones se encuentran casi siempre en las capas superiores y la capa de enlace de datos sólo realiza la tarea mínima que es "suficiente". No obstante y sin importar en dónde se encuentren, estos principios son básicamente los mismos. A menudo aparecen en sus formas más simples y puras en la capa de enlace de datos, lo que la convierte en un buen lugar para examinarlos a detalle.

3.1.1 Servicios proporcionados a la capa de red

La función de la capa de enlace de datos es proveer servicios a la capa de red. El servicio principal es la transferencia de datos de la capa de red en la máquina de origen, a la capa de red en la máquina de destino. En la capa de red de la máquina de origen está una entidad, llamada proceso, que entrega algunos bits a la capa de enlace de datos para que los transmita al destino. La tarea de la capa de enlace de datos es transmitir los bits a la máquina de destino, de modo que se puedan entregar a la capa de red de esa máquina, como se muestra en la figura 3-2(a). La transmisión real sigue la trayectoria de la figura 3-2(b), pero es más fácil pensar en términos de dos procesos de la capa de enlace de datos que se comunican mediante un protocolo de enlace de datos. Por esta razón utilizaremos de manera implícita el modelo de la figura 3-2(a) a lo largo de este capítulo.

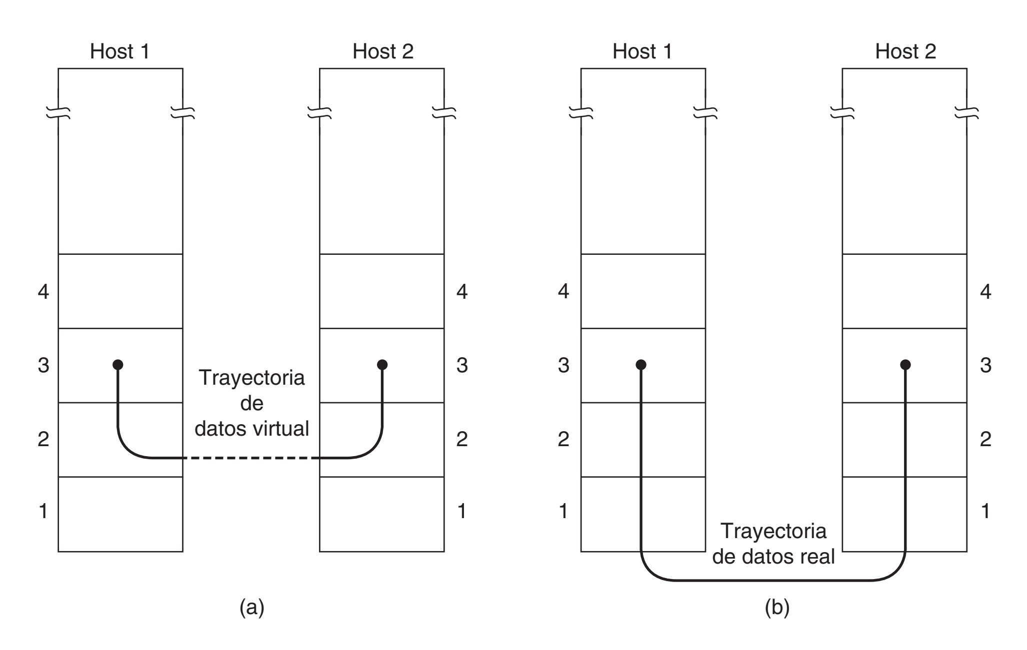


Figura 3-2. (a) Comunicación virtual. (b) Comunicación real.

La capa de enlace de datos puede diseñarse para ofrecer varios servicios. Los servicios reales ofrecidos varían de un protocolo a otro. Tres posibilidades razonables que normalmente se proporcionan son:

1. Servicio sin conexión ni confirmación de recepción.
2. Servicio sin conexión con confirmación de recepción.
3. Servicio orientado a conexión con confirmación de recepción.

El servicio sin conexión ni confirmación de recepción consiste en hacer que la máquina de origen envíe tramas independientes a la máquina de destino sin que ésta confirme la recepción. Ethernet es un buen ejemplo de una capa de enlace de datos que provee esta clase de servicio. No se establece una conexión lógica de antemano ni se libera después. Si se pierde una trama debido a ruido en la línea, en la capa de datos no se realiza ningún intento por detectar la pérdida o recuperarse de ella. Esta clase de servicio es apropiada cuando la tasa de error es muy baja, de modo que la recuperación se deja a las capas superiores. También es apropiada para el tráfico en tiempo real, como el de voz, en donde es peor tener retraso en los datos que errores en ellos.

El siguiente paso en términos de confiabilidad es el servicio sin conexión con confirmación de recepción. Cuando se ofrece este servicio tampoco se utilizan conexiones lógicas, pero se confirma de manera individual la recepción de cada trama enviada. De esta manera, el emisor sabe si la trama llegó bien o se perdió. Si no ha llegado en un intervalo especificado, se puede enviar de nuevo. Este servicio es útil en canales no confiables, como los de los sistemas inalámbricos. 802.11 (WiFi) es un buen ejemplo de esta clase de servicio.

Tal vez valga la pena enfatizar que el hecho de proporcionar confirmaciones de recepción en la capa de enlace de datos es tan sólo una optimización, nunca un requerimiento. La capa de red siempre puede enviar un paquete y esperar a que su igual en la máquina remota confirme su recepción. Si la confirmación no llega antes de que expire el temporizador, el emisor puede volver a enviar el mensaje completo. El problema con esta estrategia es que puede ser ineficiente. Por lo general los enlaces tienen una estricta longitud máxima para la trama, la cual es impuesta por el hardware, además de los retardos de propagación conocidos. La capa de red no conoce estos parámetros. Podría enviar un paquete largo que se divida,

por ejemplo, en 10 tramas, de las cuales pudieran perderse dos en promedio. Entonces se requeriría mucho tiempo para que el paquete pudiera llegar al otro extremo. Por el contrario, si las tramas se confirman y retransmiten de manera individual, entonces los errores pueden corregirse de una manera más directa y rápida. En los canales confiables, como la fibra óptica, la sobrecarga que implica el uso de un protocolo de enlace de datos muy robusto puede ser innecesaria, pero en canales inalámbricos (no confiables por naturaleza) bien vale la pena el costo.

Recapitulando el servicio más sofisticado que puede proveer la capa de enlace de datos a la capa de red es el servicio orientado a conexión. Con este servicio, las máquinas de origen y de destino establecen una conexión antes de transferir datos. Cada trama enviada a través de la conexión está numerada, y la capa de enlace de datos garantiza que cada trama enviada llegará a su destino. Es más, garantiza que cada trama se recibirá exactamente una vez y que todas las tramas se recibirán en el orden correcto. Así, el servicio orientado a conexión ofrece a los procesos de la capa de red el equivalente a un flujo de bits confiable. Es apropiado usarlo en enlaces largos y no confiables, como un canal de satélite o un circuito telefónico de larga distancia. Si se utilizara el servicio no orientado a conexión con confirmación de recepción, es posible que las confirmaciones de recepción perdidas ocasionaran que una trama se enviara y recibiera varias veces, desperdiciando ancho de banda.

Cuando se utiliza un servicio orientado a conexión, las transferencias pasan por tres fases distintas. En la primera, la conexión se establece haciendo que ambos lados inicialicen las variables y los contadores necesarios para seguir la pista de las tramas que se recibieron y las que no. En la segunda fase se transmiten una o más tramas. En la tercera y última fase, la conexión se libera al igual que las variables, los búferes y otros recursos utilizados para mantener la conexión.

3.1.2 Entramado

Para proveer servicio a la capa de red, la capa de enlace de datos debe usar el servicio que la capa física le proporciona. Lo que hace la capa física es aceptar un flujo de bits puros y tratar de entregarlo al destino. Si el canal es ruidoso, como en la mayoría de los enlaces inalámbricos y en algunos alámbricos, la capa física agregará cierta redundancia a sus señales para reducir la tasa de error de bits a un nivel tolerable. Sin embargo, no se garantiza que el flujo de bits recibido por la capa de enlace de datos esté libre de errores. Algunos bits pueden tener distintos valores y la cantidad de bits recibidos puede ser menor, igual o mayor que la cantidad de bits transmitidos. Es responsabilidad de la capa de enlace de datos detectar y, de ser necesario, corregir los errores.

El método común es que la capa de enlace de datos divida el flujo de bits en tramas discretas, calcule un token corto conocido como suma de verificación para cada trama, e incluya esa suma de verificación en la trama al momento de transmitirla (más adelante en este capítulo analizaremos los algoritmos de suma de verificación). Cuando una trama llega al destino, se recalcula la suma de verificación. Si la nueva suma de verificación calculada es distinta de la contenida en la trama, la capa de enlace de datos sabe que ha ocurrido un error y toma las medidas necesarias para manejarlo (por ejemplo, desecha la trama errónea y es posible que también devuelva un informe de error).

Es más difícil dividir el flujo de bits en tramas de lo que parece a simple vista. Un buen diseño debe facilitar a un receptor el proceso de encontrar el inicio de las nuevas tramas al tiempo que utiliza una pequeña parte del ancho de banda del canal. En esta sección veremos cuatro métodos:

1. Conteo de bytes.
2. Bytes bandera con relleno de bytes.
3. Bits bandera con relleno de bits.
4. Violaciones de codificación de la capa física.

El primer método de entramado se vale de un campo en el encabezado para especificar el número de bytes en la trama. Cuando la capa de enlace de datos del destino ve el conteo de bytes, sabe cuántos bytes siguen y, por lo tanto, dónde concluye la trama. Esta técnica se muestra en la figura 3-3(a) para cuatro tramas pequeñas de ejemplo con 5, 5, 8 y 8 bytes de longitud, respectivamente.

El problema con este algoritmo es que el conteo se puede alterar debido a un error de transmisión. Por ejemplo, si el conteo de bytes de 5 en la segunda trama de la figura 3-3(b) se vuelve un 7 debido a que cambió un solo bit, el destino perderá la sincronía y entonces será incapaz de localizar el inicio correcto de la siguiente trama. Incluso si el destino sabe que la trama está mal puesto que la suma de verificación es incorrecta, no tiene forma de saber dónde comienza la siguiente trama. Tampoco es útil enviar una trama de vuelta a la fuente para solicitar una retransmisión, ya que el destino no sabe cuántos bytes tiene que saltar para llegar al inicio de la retransmisión. Por esta razón raras veces se utiliza el método de conteo de bytes por sí solo.

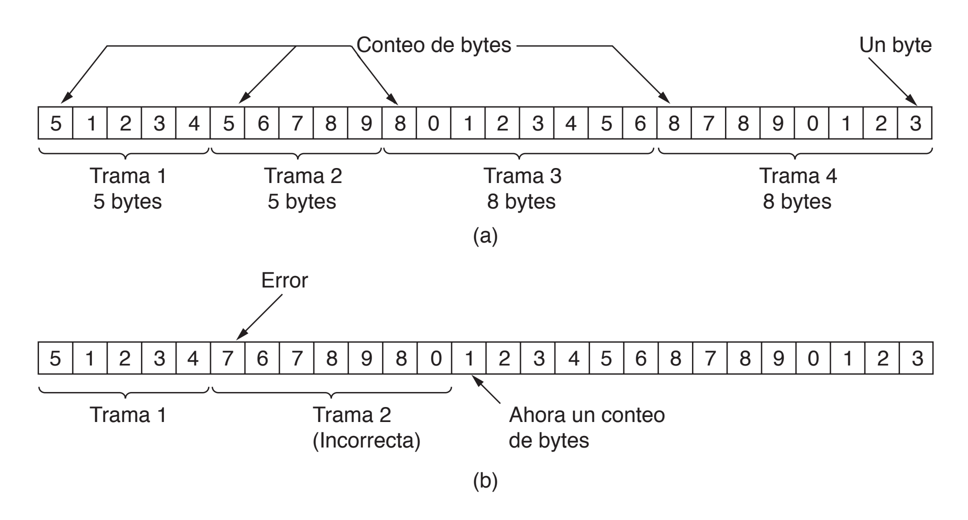


Figura 3-3. Un flujo de bytes. (a) Sin errores. (b) Con un error.

El segundo método de entramado evita el problema de volver a sincronizar nuevamente después de un error al hacer que cada trama inicie y termine con bytes especiales. Con frecuencia se utiliza el mismo byte, denominado **byte bandera**, como delimitador inicial y final. Este byte se muestra en la figura 3-4(a) como FLAG. Dos bytes bandera consecutivos señalan el final de una trama y el inicio de la siguiente. De esta forma, si el receptor pierde alguna vez la sincronización, todo lo que tiene que hacer es buscar dos bytes bandera para encontrar el fin de la trama actual y el inicio de la siguiente.

Sin embargo, aún queda un problema que tenemos que resolver. Se puede dar el caso de que el byte bandera aparezca en los datos, en especial cuando se transmiten datos binarios como fotografías o canciones. Esta situación interferiría con el entramado. Una forma de resolver este problema es hacer que la capa de enlace de datos del emisor inserte un byte de escape especial (ESC) justo antes de cada byte bandera "accidental" en los datos. De esta forma es posible diferenciar un byte bandera del entramado de uno en los datos mediante la ausencia o presencia de un byte de escape antes del byte bandera. La capa de enlace de datos del lado receptor quita el byte de escape antes de entregar los datos a la capa de red. Esta técnica se llama **relleno de bytes**.

Ahora bien, la siguiente pregunta es: ¿qué sucede si aparece un byte de escape en medio de los datos? La respuesta es que también se rellena con un byte de escape. En el receptor se quita el primer byte de escape y se deja el byte de datos que le sigue (el cual podría ser otro byte de escape, o incluso el byte ban-

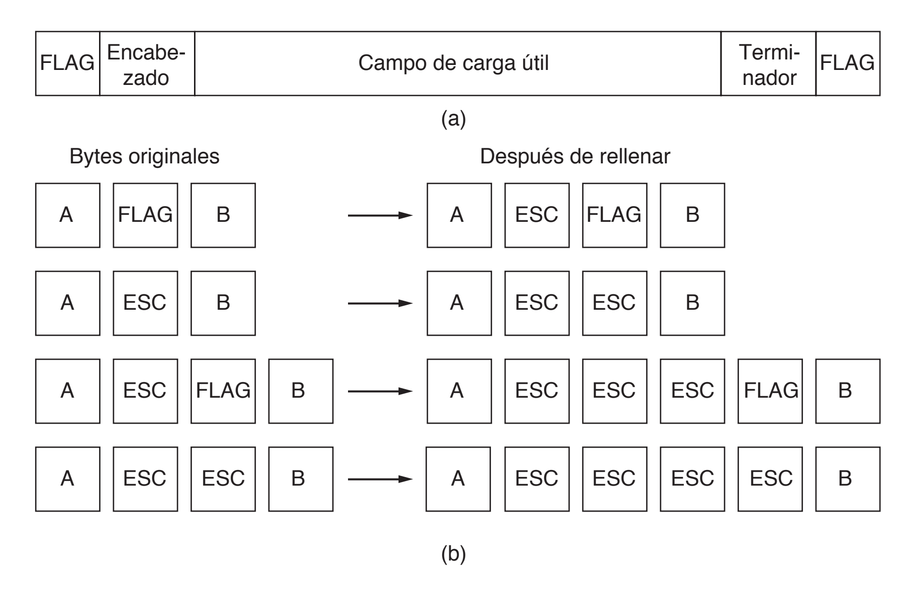


Figura 3-4. (a) Una trama delimitada por bytes bandera. (b) Cuatro ejemplos de secuencias de bytes antes y después del relleno de bytes.

dera). En la figura 3-4(b) se muestran algunos ejemplos. En todos los casos, la secuencia de bytes que se entrega después de quitar los bytes de relleno es exactamente la misma que la secuencia de bytes original. Así todavía podemos encontrar un límite de trama si buscamos dos bytes bandera seguidos, sin molestarnos por eliminar los escapes.

El esquema de relleno de bytes que se muestra en la figura 3-4 es una ligera simplificación del esquema empleado en el protocolo **PPP** (**Protocolo Punto a Punto**, del inglés *Point-to-Point Protocol*), que se utiliza para transmitir paquetes a través de los enlaces de comunicación. Analizaremos el protocolo PPP casi al final de este capítulo.

El tercer método de delimitar el flujo de bits resuelve una desventaja del relleno de bytes: que está obligado a usar bytes de 8 bits. También se puede realizar el entramado a nivel de bit, de modo que las tramas puedan contener un número arbitrario de bits compuesto por unidades de cualquier tamaño. Esto se desarrolló para el protocolo **HDLC** (**Control de Enlace de Datos de Alto Nivel**, del inglés *High-level Data Link Control*), que alguna vez fue muy popular. Cada trama empieza y termina con un patrón de bits especial, 01111110 o 0x7E en hexadecimal. Este patrón es un byte bandera. Cada vez que la capa de enlace de datos del emisor encuentra cinco bits 1 consecutivos en los datos, inserta automáticamente un 0 como relleno en el flujo de bits de salida. Este **relleno de bits** es análogo al relleno de bytes, en el cual se inserta un byte de escape en el flujo de caracteres de salida antes de un byte bandera en los datos. Además asegura una densidad mínima de transiciones que ayudan a la capa física a mantener la sincronización. La tecnología USB (Bus Serie Universal, del inglés *Universal Serial Bus*) usa relleno de bits por esta razón.

Cuando el receptor ve cinco bits 1 de entrada consecutivos, seguidos de un bit 0, extrae (es decir, borra) de manera automática el bit 0 de relleno. Así como el relleno de bytes es completamente transparente para la capa de red en ambas computadoras, también lo es el relleno de bits. Si los datos de usuario contienen el patrón bandera 01111110, éste se transmite como 011111010, pero se almacena en la memoria del receptor como 01111110. En la figura 3-5 se muestra un ejemplo del relleno de bits.

Con el relleno de bits, el límite entre las dos tramas puede ser reconocido sin ambigüedades mediante el patrón bandera. De esta manera, si el receptor pierde la pista de dónde está, todo lo que tiene que hacer

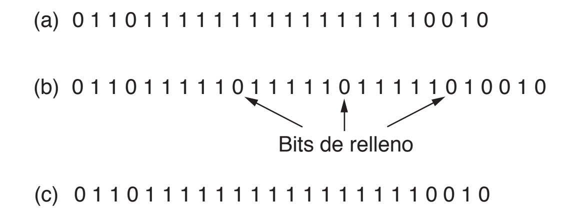


Figura 3-5. Relleno de bits. (a) Los datos originales. (b) Los datos, según aparecen en la línea. (c) Los datos, como se almacenan en la memoria del receptor después de quitar el relleno.

es explorar la entrada en busca de secuencias de banderas, pues sólo pueden ocurrir en los límites de las tramas y nunca dentro de los datos.

Un efecto secundario del relleno de bits y de bytes es que la longitud de una trama depende ahora del contenido de los datos que lleva. Por ejemplo, si no hay bytes bandera en los datos, se podrían llevar 100 bytes en una trama de aproximadamente 100 bytes. No obstante, si los datos consisten sólo de bytes bandera, habrá que incluir un byte escape para cada uno de estos bytes y la trama será de cerca de 200 bytes de longitud. Con el relleno de bits, el aumento sería cerca del 12.5%, ya que se agrega 1 bit a cada byte.

El último método de entramado es utilizar un atajo desde la capa física. En el capítulo 2 vimos que la codificación de bits como señales incluye a menudo redundancia para ayudar al receptor. Esta redundancia significa que algunas señales no ocurrirán en los datos regulares. Por ejemplo, en el código de línea 4B/5B se asignan 4 bits de datos a 5 bits de señal para asegurar suficientes transiciones de bits. Esto significa que no se utilizan 16 de las 32 posibles señales. Podemos usar algunas señales reservadas para indicar el inicio y el fin de las tramas. En efecto, estamos usando "violaciones de código" para delimitar tramas. La belleza de este esquema es que, como hay señales reservadas, es fácil encontrar el inicio y final de las tramas y no hay necesidad de rellenar los datos.

Muchos protocolos de enlace de datos usan una combinación de estos métodos por seguridad. Un patrón común utilizado para Ethernet y 802.11 es hacer que una trama inicie con un patrón bien definido, conocido como **preámbulo**. Este patrón podría ser bastante largo (es común que cuente con 72 bits para 802.11) de modo que el receptor se pueda preparar para un paquete entrante. El preámbulo va seguido de un campo de longitud (cuenta) en el encabezado, que se utiliza para localizar el final de la trama.

3.1.3 Control de errores

Una vez resuelto el problema de marcar el inicio y el fin de cada trama, llegamos al siguiente dilema: cómo asegurar que todas las tramas realmente se entreguen en el orden apropiado a la capa de red del destino. Suponga por un momento que el receptor puede saber si una trama que recibe contiene la información correcta o errónea (en la sección 3.2 analizaremos los códigos que se utilizan para detectar y corregir errores de transmisión). Para un servicio sin conexión ni confirmación de recepción sería ideal si el emisor siguiera enviando tramas sin importarle si llegan en forma adecuada. Pero para un servicio confiable orientado a conexión no sería nada bueno.

La manera normal de asegurar la entrega confiable de datos es proporcionar retroalimentación al emisor sobre lo que está ocurriendo en el otro lado de la línea. Por lo general, el protocolo exige que el receptor devuelva tramas de control especiales que contengan confirmaciones de recepción positivas o negativas de las tramas que llegan. Si el emisor recibe una confirmación de recepción positiva de una trama, sabe que la trama llegó de manera correcta. Por otra parte, una confirmación de recepción negativa significa que algo falló y que se debe transmitir la trama otra vez.

Una complicación adicional surge de la posibilidad de que los problemas de hardware causen la desaparición de una trama completa (por ejemplo, por una ráfaga de ruido). En este caso, el receptor no reaccionará en absoluto, ya que no tiene razón para reaccionar. De manera similar, si se pierde la trama de confirmación de recepción, el emisor no sabrá cómo proceder. Debe quedar claro que en un protocolo en el cual el emisor envía una trama y luego espera una confirmación de recepción, positiva o negativa, éste se quedaría esperando eternamente si se perdiera por completo una trama debido a, por ejemplo, una falla de hardware o un canal de comunicación defectuoso.

Para manejar esta posibilidad se introducen temporizadores en la capa de enlace de datos. Cuando el emisor envía una trama, por lo general también inicia un temporizador. Éste se ajusta de modo que expire cuando haya transcurrido un intervalo suficiente para que la trama llegue a su destino, se procese ahí y la confirmación de recepción se propague de vuelta al emisor. Por lo general, la trama se recibirá de manera correcta y la confirmación de recepción llegará antes de que el temporizador expire, en cuyo caso se cancelará.

No obstante, si la trama o la confirmación de recepción se pierde, el temporizador expirará, alertando al emisor sobre un problema potencial. La solución obvia es simplemente transmitir de nuevo la trama. Sin embargo, aunque éstas pueden transmitirse muchas veces, existe el peligro de que el receptor acepte la misma trama en dos o más ocasiones y que la pase a la capa de red más de una vez. Para evitar que esto ocurra, generalmente es necesario asignar números de secuencia a las tramas de salida, con el fin de que el receptor pueda distinguir las retransmisiones de las originales.

El asunto de la administración de temporizadores y números de secuencia para asegurar que cada trama llegue finalmente a la capa de red en el destino una sola vez, ni más ni menos, es una parte importante de las tareas de la capa de enlace de datos (y de las capas superiores). Más adelante en este capítulo estudiaremos la manera en que se lleva a cabo esta administración, mediante la observación de una serie de ejemplos cada vez más sofisticados.

3.1.4 Control de flujo

Otro tema de diseño importante que se presenta en la capa de enlace de datos (y también en las capas superiores) es qué hacer con un emisor que quiere transmitir tramas de manera sistemática y a mayor velocidad que aquella con que puede aceptarlos el receptor. Esta situación puede ocurrir cuando el emisor opera en una computadora rápida y el receptor trabaja en una máquina lenta. Una situación común es cuando un teléfono inteligente solicita una página web de un servidor mucho más poderoso, que a su vez enciende la manguera de bomberos y dispara el chorro de datos al pobre e indefenso teléfono hasta que queda totalmente saturado. Aunque la transmisión esté libre de errores, en cierto punto el receptor simplemente no será capaz de manejar las tramas conforme lleguen y comenzará a perder algunas.

Es obvio que tiene que hacerse algo para evitar esta situación. Por lo general, se utilizan dos métodos. En el primero, el **control de flujo basado en retroalimentación**, el receptor regresa información al emisor para autorizarle que envíe más datos o por lo menos indicarle su estado. En el segundo, el **control de flujo basado en tasa**, el protocolo tiene un mecanismo integrado que limita la tasa a la que el emisor puede transmitir los datos, sin recurrir a la retroalimentación por parte del receptor.

En este capítulo estudiaremos los esquemas de control de flujo basados en retroalimentación, principalmente debido a que los esquemas basados en tasa sólo se ven como parte de la capa de transporte (capítulo 5). Los esquemas basados en retroalimentación se ven tanto en la capa de enlace como en las capas superiores. En realidad, es más común esto último, en cuyo caso el hardware de la capa de enlace se diseña para operar con la rapidez suficiente como para no producir pérdidas. Por ejemplo, se dice algunas veces que las implementaciones de hardware de la capa de enlace como **NIC** (**Tarjetas de Interfaz de Red**, del inglés *Network Interface Cards*) operan a "velocidad de alambre", lo cual significa que pueden

manejar las tramas con la misma rapidez con que pueden llegar al enlace. De esta forma, los excesos no son problema del enlace, por lo que se manejan en las capas superiores.

Se conocen varios esquemas de control de flujo basados en retroalimentación, pero la mayoría se basa en el mismo principio. El protocolo contiene reglas bien definidas respecto al momento en que un emisor puede transmitir la siguiente trama. Con frecuencia estas reglas prohíben el envío de tramas hasta que el receptor lo autorice, ya sea en forma implícita o explícita. Por ejemplo, cuando se establece una conexión, el receptor podría decir: "Puedes enviarme *n* tramas ahora, pero una vez que lo hagas, no envíes nada más hasta que te indique que continúes". Más adelante analizaremos los detalles.

3.2 DETECCIÓN Y CORRECCIÓN DE ERRORES

En el capítulo 2 vimos que los canales de comunicación tienen una variedad de características. Algunos de ellos, como la fibra óptica en las redes de telecomunicaciones, tienen tasas de error pequeñas de modo que los errores de transmisión son una rara ocurrencia. Pero otros canales, en especial los enlaces inalámbricos y los viejos lazos locales, tienen tasas de error más grandes. Para estos enlaces, los errores de transmisión son la norma. No se pueden evitar a un costo razonable en términos de rendimiento. La conclusión es que los errores de transmisión están aquí para quedarse. Tenemos que aprender a lidiar con ellos.

Los diseñadores de redes han desarrollado dos estrategias básicas para manejar los errores. Ambas añaden información redundante a los datos que se envían. Una es incluir suficiente información redundante para que el receptor pueda deducir cuáles debieron ser los datos transmitidos. La otra estrategia es incluir sólo suficiente redundancia para permitir que el receptor sepa que ha ocurrido un error (pero no qué error) y entonces solicite una retransmisión. La primera estrategia utiliza **códigos de corrección de errores**; la segunda usa **códigos de detección de errores**. El uso de códigos de corrección de errores por lo regular se conoce como **FEC** (**Corrección de Errores hacia Adelante**, del inglés *Forward Error Correction*).

Cada una de estas técnicas ocupa un nicho ecológico diferente. En los canales que son altamente confiables, como los de fibra, es más económico utilizar un código de detección de errores y sólo retransmitir los bloques defectuosos que surgen ocasionalmente. Sin embargo, en los canales que causan muchos errores, como los enlaces inalámbricos, es mejor agregar la redundancia suficiente a cada bloque para que el receptor pueda descubrir cuál era el bloque original que se transmitió. La FEC se utiliza en canales ruidosos puesto que las retransmisiones tienen la misma probabilidad de ser tan erróneas como la primera transmisión.

Una consideración clave para estos códigos es el tipo de errores que pueden llegar a ocurrir. Ni los códigos de corrección de errores ni los de detección de errores pueden manejar todos los posibles errores, puesto que los bits redundantes que ofrecen protección tienen la misma probabilidad de ser recibidos con errores que los bits de datos (lo cual puede comprometer su protección). Sería ideal que el canal tratara a los bits redundantes de una manera distinta a los bits de datos, pero no es así. Para el canal todos son sólo bits. Esto significa que para evitar errores no detectados, el código debe ser lo bastante robusto como para manejar los errores esperados.

Un modelo es que los errores son producidos por valores extremos de ruido térmico que satura la señal breve y ocasionalmente, lo cual produce errores aislados de un solo bit. Otro modelo es que los errores tienden a producirse en ráfagas en vez de hacerlo en forma individual. Este modelo se deriva de los procesos físicos que los generan (como un desvanecimiento pronunciado en un canal inalámbrico o una interferencia eléctrica transitoria en un canal alámbrico).

Ambos modelos importan en la práctica, además de que tienen distintas concesiones. El hecho de que los errores lleguen en ráfagas tiene tanto ventajas como desventajas en comparación con los errores aislados de un solo bit. Por el lado positivo, los datos de computadora siempre se envían en bloques de

bits. Suponga que el tamaño de bloque es de 1000 bits y que la tasa de error es de 0.001 por bit. Si los errores fueran independientes, la mayoría de los bloques contendría un error. Pero si los errores llegaran en ráfagas de 100, sólo un bloque de esos 100 sería afectado en promedio. La desventaja de los errores en ráfaga es que cuando ocurren son mucho más difíciles de corregir que los errores aislados.

También existen otros tipos de errores. Algunas veces se conocerá la ubicación de un error, tal vez debido a que la capa física recibió una señal analógica que estaba muy alejada del valor esperado para un 0 o un 1 y declaró el bit como perdido. A esta situación se le conoce como **canal de borrado**. Es más fácil corregir los errores en los canales de borrado que en canales que voltean bits, ya que incluso si se pierde el valor del bit, por lo menos sabemos cuál tiene el error. Sin embargo, algunas veces no tenemos el beneficio de los canales de borrado.

A continuación examinaremos los códigos de corrección de errores y los códigos de detección de errores. Pero debe tener en cuenta dos puntos. Primero, cubrimos estos códigos en la capa de enlace debido a que es el primer lugar en el que nos topamos con el problema de transmitir grupos de bits de manera confiable. Sin embargo, los códigos se utilizan ampliamente debido a que la confiabilidad es una preocupación general. Los códigos de corrección de errores se ven también en la capa física, en especial con los canales ruidosos, y en capas más altas, en especial con los medios de tiempo real y la distribución de contenido. Los códigos de detección de errores se utilizan con frecuencia en las capas de enlace, red y transporte.

El segundo punto a considerar es que los códigos de error son matemáticas aplicadas. A menos que usted sea muy adepto a los campos de Galois o a las propiedades de las matrices dispersas, es más conveniente que obtenga códigos con buenas propiedades de una fuente confiable en vez de crear sus propios códigos. De hecho, esto es lo que hacen muchos estándares de protocolos, en donde los mismos códigos se utilizan una y otra vez. En el material que veremos a continuación, estudiaremos con detalle un código simple y después describiremos brevemente los códigos avanzados. De esta forma podremos comprender las concesiones a través del código simple y hablaremos sobre los códigos que se utilizan en la práctica a través de los códigos avanzados.

3.2.1 Códigos de corrección de errores

Analizaremos cuatro códigos de corrección de errores:

1. Códigos de Hamming.
2. Códigos convolucionales binarios.
3. Códigos de Reed-Solomon.
4. Códigos de verificación de paridad de baja densidad.

Todos estos códigos agregan redundancia a la información que se envía. Una trama consiste en m bits de datos (mensaje) y r bits redundantes (verificación). En un **código de bloque**, los r bits de verificación se calculan únicamente en función de los m bits de datos con los que se asocian, como si los m bits se buscaran en una gran tabla para encontrar sus correspondientes r bits de verificación. En un **código sistemático**, los m bits de datos se envían directamente, junto con los bits de verificación, en vez de que se codifiquen por sí mismos antes de enviarlos. En un **código lineal**, los r bits de verificación se calculan como una función lineal de los m bits de datos. El OR exclusivo (XOR) o la suma de módulo 2 es una elección popular. Esto significa que la codificación se puede llevar a cabo con operaciones como multiplicaciones de matrices o circuitos lógicos simples. Los códigos que analizaremos en esta sección son códigos de bloque lineales sistemáticos, a menos que se indique otra cosa.

Sea la longitud total de un bloque n (es decir, $n = m + r$). Describiremos esto como un código (n, m). Una unidad de n bits que contiene bits de datos y de verificación se conoce como **palabra codificada**

de n bits. La **tasa de código**, o simplemente tasa, es la fracción de la palabra codificada que lleva información no redundante, o m/n. Las tasas que se utilizan en la práctica varían mucho. Podrían ser 1/2 para un canal ruidoso, en cuyo caso la mitad de la información recibida es redundante, o podrían estar cerca de 1 para un canal de alta calidad, en donde sólo se agrega un pequeño número de bits de verificación a un mensaje extenso.

Para entender la manera en que pueden manejarse los errores, es necesario estudiar de cerca lo que es en realidad un error. Dadas dos palabras codificadas cualesquiera que se pueden transmitir o recibir, digamos 10001001 y 10110001, es posible determinar cuántos bits correspondientes difieren. En este caso, difieren 3 bits. Para determinar la cantidad de bits diferentes, basta aplicar un XOR a las dos palabras codificadas y contar la cantidad de bits 1 en el resultado, por ejemplo:

$$\begin{array}{r} 10001001 \\ \underline{10110001} \\ 00111000 \end{array}$$

La cantidad de posiciones de bits en la que difieren dos palabras codificadas se llama **distancia de Hamming** (Hamming, 1950). Su significado es que, si dos palabras codificadas están separadas una distancia de Hamming d, se requerirán d errores de un solo bit para convertir una en la otra.

Dado el algoritmo para calcular los bits de verificación, es posible construir una lista completa de las palabras codificadas válidas, y a partir de esta lista se pueden encontrar las dos palabras codificadas con la menor distancia de Hamming. Esta distancia es la distancia de Hamming del código completo.

En la mayoría de las aplicaciones de transmisión de datos, todos los 2^m mensajes de datos posibles son válidos, pero debido a la manera en que se calculan los bits de verificación no se usan todas las 2^n palabras codificadas posibles. De hecho, cuando hay r bits de verificación sólo la pequeña fracción de $2^m/2^n$ o $1/2^r$ de los posibles mensajes serán palabras codificadas válidas. Esta dispersión con la que se incrusta el mensaje en el espacio de las palabras codificadas es la que permite que el receptor detecte y corrija los errores.

Las propiedades de detección y corrección de errores de un código de bloque dependen de su distancia de Hamming. Para detectar d errores de manera confiable se necesita un código con distancia $d + 1$, pues con tal código no hay manera de que d errores de un bit puedan cambiar una palabra codificada válida a otra. Cuando el receptor ve una palabra codificada inválida, sabe que ha ocurrido un error de transmisión. De manera similar, para corregir d errores se necesita un código de distancia $2d + 1$, pues así las palabras codificadas válidas están tan separadas que, aun con d cambios, la palabra codificada original sigue estando más cercana que cualquier otra. Esto significa que la palabra codificada original se puede determinar en forma única con base en la suposición de que es menos probable un mayor número de errores.

Como ejemplo sencillo de un código de corrección de errores, considere un código con sólo cuatro palabras codificadas válidas:

0000000000, 0000011111, 1111100000 y 1111111111

Este código tiene una distancia de 5, lo cual significa que puede corregir errores dobles o detectar errores cuádruples. Si llega la palabra codificada 0000000111 y esperamos sólo errores de uno o dos bits, el receptor sabrá que la palabra original debió haber sido 0000011111. Pero si un error triple cambia 0000000000 a 0000000111, el error no se corregirá en forma apropiada. Por otro lado, si esperamos todos estos errores, podremos detectarlos. Ninguna de las palabras codificadas recibidas es válida, por lo que debe haber ocurrido un error. Debe ser aparente que en este ejemplo no podemos corregir errores dobles y detectar al mismo tiempo errores cuádruples, ya que tendríamos que interpretar de dos maneras distintas una palabra codificada recibida.

En nuestro ejemplo, la tarea de decodificar mediante el proceso de buscar la palabra codificada válida que se asemeje más a la palabra codificada recibida se puede llevar a cabo mediante inspección. Por desgracia, en el caso más general en donde hay que evaluar todas las palabras codificadas como candidatas, esta tarea puede requerir de mucho tiempo. Como alternativa se han diseñado códigos prácticos que nos permiten usar atajos para buscar cuál tendría más probabilidades de ser la palabra codificada original.

Imagine que deseamos diseñar un código con *m* bits de mensaje y *r* bits de verificación que permitirá la corrección de todos los errores individuales. Cada uno de los 2^m mensajes válidos tiene *n* palabras codificadas inválidas a una distancia 1 de él. Éstas se forman invirtiendo de manera sistemática cada uno de los *n* bits de la palabra codificada de *n* bits que la conforman. Por lo tanto, cada uno de los 2^m mensajes válidos requiere $n + 1$ patrones de bits dedicados a él. Dado que la cantidad total de patrones de bits es 2^n, debemos tener $(n + 1)2^m \leq 2^n$. Si usamos $n + m + r$, este requisito se vuelve

$$(m + r + 1) \leq 2^r \quad (3\text{-}1)$$

Dado el valor de *m*, esto impone un límite inferior a la cantidad de bits de verificación necesarios para corregir errores individuales.

De hecho, este límite inferior teórico puede lograrse mediante el uso de un método desarrollado por Hamming (1950). En los **códigos de Hamming**, los bits de la palabra codificada se numeran en forma consecutiva, comenzando por el bit 1 a la izquierda, el bit 2 a su derecha inmediata, etc. Los bits que son potencias de 2 (1, 2, 4, 8, 16, etc.) son bits de verificación. El resto (3, 5, 6, 7, 9, etc.) se rellenan con los *m* bits de datos. El patrón se muestra para un código de Hamming (11,7) con 7 bits de datos y 4 bits de verificación en la figura 3-6. Cada bit de verificación obliga a que la suma módulo 2, o paridad de un grupo de bits, incluyéndolo a él mismo, sea par (o impar). Un bit puede estar incluido en varios cálculos de bits de verificación. Para ver a qué bits de verificación contribuye el bit de datos en la posición *k*, reescriba *k* como una suma de potencias de 2. Por ejemplo, $11 = 1 + 2 + 8$ y $29 = 1 + 4 + 8 + 16$. Un bit es verificado solamente por los bits de verificación que ocurren en su expansión (por ejemplo, el bit 11 es verificado por los bits 1, 2 y 8). En el ejemplo se calculan los bits de verificación para las sumas con paridad par de un mensaje que contiene la letra “A” del código ASCII.

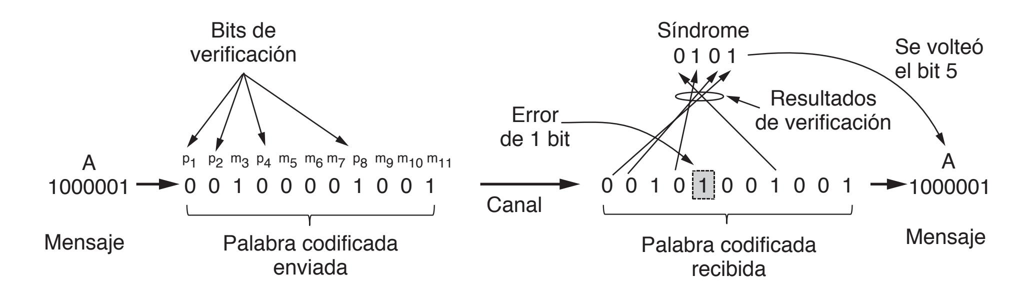


Figura 3-6. Ejemplo de un código de Hamming (11,7) para corregir errores de un solo bit.

Esta construcción produce un código con una distancia de Hamming de 3, lo cual significa que puede corregir errores individuales (o detectar errores dobles). La razón de enumerar con mucho cuidado el mensaje y los bits de verificación se vuelve aparente en el proceso de decodificación. Cuando llega una palabra codificada, el receptor vuelve a realizar los cálculos del bit de verificación, incluyendo los valores de los bits de verificación recibidos. A estos les llamamos resultados de verificación. Si los bits de veri-

ficación están correctos, entonces cada resultado de verificación debe ser cero para las sumas de paridad par. En este caso la palabra codificada se acepta como válida.

Pero si no todos los resultados de verificación son cero, entonces se ha detectado un error. El conjunto de resultados de verificación forma el **síndrome de error** que se utiliza para señalarlo y corregirlo el error. En la figura 3-6 ocurrió un error de un solo bit en el canal, de modo que los resultados de verificación son 0, 1, 0 y 1 para $k = 8, 4, 2$ y 1, respectivamente. Esto nos da un síndrome de 0101, o $4 + 1 = 5$. Según el diseño del esquema, esto significa que el quinto bit es un error. Al voltear el bit incorrecto (que podría ser un bit de verificación o uno de datos) y desechar los bits de verificación obtenemos el mensaje correcto de una letra "A" en código ASCII.

Las distancias de Hamming son valiosas para comprender los códigos de bloque, y los códigos de Hamming se utilizan en la memoria de corrección de errores. Sin embargo, la mayoría de las redes utilizan códigos más robustos. El segundo código que veremos es el **código convolucional**. Este código es el único que analizaremos que no es código de bloque. En un código convolucional, un codificador procesa una secuencia de bits de entrada y genera una secuencia de bits de salida. No hay un tamaño de mensaje natural, o límite de codificación, como en un código de bloque. La salida depende de los bits de entrada actual y previa. Es decir, el codificador tiene memoria. El número de bits previos de los que depende la salida se denomina **longitud de restricción** del código. Los códigos convolucionales se especifican en términos de su tasa de transmisión y su longitud de restricción.

Los códigos convolucionales se utilizan mucho en las redes implementadas; por ejemplo, como parte del sistema de telefonía móvil GSM, en las comunicaciones de satélite y en 802.11. Como ejemplo, en la figura 3-7 se muestra un código convolucional popular. Este código se conoce como código convolucional NASA de $r = 1/2$ y $k = 7$, ya que se utilizó por primera vez en las misiones espaciales del Voyager a partir de 1977. Desde entonces se ha reutilizado libremente, por ejemplo, como parte de las redes 802.11.

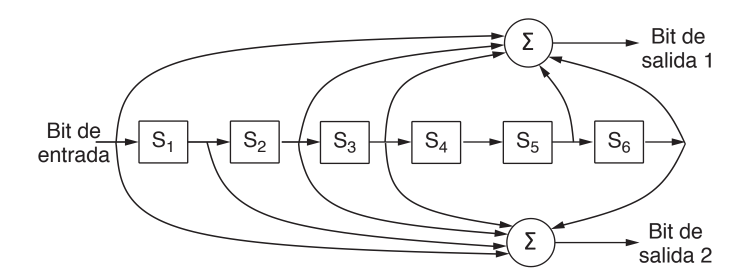


Figura 3-7. El código convolucional binario NASA utilizado en redes 802.11.

En la figura 3-7, cada bit de entrada del lado izquierdo produce dos bits de salida del lado derecho, los cuales son las sumas XOR de la entrada y el estado interno. Ya que se trata con bits y se realizan operaciones lineales, es un código convolucional binario lineal. Puesto que 1 bit de entrada produce 2 bits de salida, la tasa de código es de 1/2. No es sistemático, ya que ninguno de los bits de salida es simplemente el bit de entrada.

El estado interno se mantiene en seis registros de memoria. Cada vez que se introduce otro bit, los valores de los registros se desplazan a la derecha. Por ejemplo, si se introduce 111 como entrada y el estado inicial está compuesto sólo de ceros, entonces el estado interno (que se escribe de izquierda a derecha) se convertirá en 100000, 110000 y 111000 después de haber introducido el primer, segundo y tercer bits, respectivamente. Los bits de salida serán 11, seguidos de 10 y después de 01. Se requieren siete desplazamientos para vaciar una entrada por completo, de modo que no afecte la salida. Por lo tanto, la longitud de restricción de este código es $k = 7$.

Para decodificar un código convolucional es necesario buscar la secuencia de bits de entrada que tenga la mayor probabilidad de haber producido la secuencia observada de bits de salida (incluyendo los errores). Para valores pequeños de k esto se hace mediante un algoritmo muy popular desarrollado por Viterbi (Forney, 1973). El algoritmo recorre la secuencia observada y guarda para cada paso y cada posible estado interno la secuencia de entrada que hubiera producido la secuencia observada con la menor cantidad posible de errores. Al final, la secuencia de entrada que requiera la menor cantidad de errores es el mensaje más probable.

Los códigos convolucionales han sido populares en la práctica, ya que es fácil ignorar la incertidumbre de que un bit sea 0 o 1 en la decodificación. Por ejemplo, suponga que -1 V es el nivel 0 lógico y $+1$ V es el nivel 1 lógico, y que podríamos recibir 0.9 V y -0.1V para 2 bits. En vez de asignar estas señales a 1 y 0 de inmediato, podría ser conveniente considerar que 0.9 V "muy probablemente sea un 1" y que -0.1 V "puede ser un 0", y corregir la secuencia en su totalidad. Las extensiones del algoritmo de Viterbi pueden trabajar con estas incertidumbres para ofrecer una corrección de errores más poderosa. Este método de trabajar con la incertidumbre de un bit se conoce como **decodificación de decisión suave**. Por el contrario, decidir si cada bit es un 0 o un 1 antes de la subsiguiente corrección de errores se conoce como **decodificación de decisión dura**.

El tercer tipo de código de corrección de errores que describiremos es el **código de Reed-Solomon**. Al igual que los códigos de Hamming, los códigos de Reed-Solomon son códigos de bloques lineales y con frecuencia también son sistemáticos. A diferencia de los códigos de Hamming, que operan sobre bits individuales, los códigos de Reed-Solomon operan sobre símbolos de m bits. Naturalmente las matemáticas son más complejas, por lo que describiremos su operación mediante una analogía.

Los códigos de Reed-Solomon se basan en el hecho de que todo polinomio de n grados se determina en forma única mediante $n + 1$ puntos. Por ejemplo, una línea con la forma $ax + b$ se determina mediante dos puntos. Los puntos extra en la misma línea son redundantes, lo cual es útil para la corrección de errores. Imagine que tenemos dos puntos de datos que representan una línea y que enviamos esos dos puntos de datos junto con dos puntos de verificación seleccionados sobre la misma línea. Si uno de los puntos se recibe con error, de todas formas podemos recuperar los puntos de datos si ajustamos una línea a los puntos recibidos. Tres de los puntos estarán en la línea y el otro punto (el del error) no. Al encontrar la línea hemos corregido el error.

En realidad los códigos de Reed-Solomon se definen como polinomios que operan sobre campos finitos, pero trabajan de una manera similar. Para símbolos de m bits, las palabras codificadas son de $2^m - 1$ símbolos de longitud. Una elección popular es hacer a $m = 8$, de modo que los símbolos sean bytes. Así, una palabra codificada tiene una longitud de 256 bytes. El código (255, 233) se utiliza ampliamente; agrega 32 símbolos redundantes a 233 símbolos de datos. La decodificación con corrección de errores se realiza mediante un algoritmo desarrollado por Berlekamp y Massey, el cual puede realizar con eficiencia la tarea de ajuste para los códigos de longitud moderada (Massey, 1969).

Los códigos de Reed-Solomon se utilizan mucho en la práctica debido a sus poderosas propiedades de corrección de errores, en especial para los errores de ráfagas. Se utilizan para DSL, datos sobre cable, comunicaciones de satélite y tal vez de manera más ubicua en los CD, DVD y discos Blu-ray. Puesto que se basan en símbolos de m bits, tanto un error de un solo bit como un error de ráfaga de m bits se tratan simplemente como error de un símbolo. Cuando se agregan $2t$ símbolos redundantes, un código de Reed-Solomon es capaz de corregir hasta t errores en cualquiera de los símbolos transmitidos. Esto significa que, por ejemplo, el código (255, 233) que tiene 32 símbolos redundantes puede corregir errores de hasta 16 símbolos. Como los símbolos pueden ser consecutivos y cada uno de ellos es de 8 bits, se puede corregir una ráfaga de errores de hasta 128 bits. La situación es aún mejor si el modelo de error es el de borrado (por ejemplo, una rayadura en un CD que borra algunos símbolos). En este caso se pueden corregir hasta $2t$ errores.

A menudo los códigos de Reed-Solomon se utilizan en combinación con otros códigos, como el convolucional. El razonamiento es el siguiente. Los códigos convolucionales son efectivos a la hora de manejar errores de bits aislados, pero es probable que fallen con una ráfaga de errores, si hay demasiados errores en el flujo de bits recibido. Al agregar un código de Reed-Solomon dentro del código convolucional, la decodificación de Reed-Solomon puede limpiar las ráfagas de errores, una tarea que realiza con mucha eficiencia. Así, el código en general provee una buena protección contra los errores individuales y los errores de ráfaga.

El último código de corrección de errores que estudiaremos es el código **LDPC** (**Verificación de Paridad de Baja Densidad**, del inglés *Low-Density Parity Check*). Los códigos LDPC son códigos de bloques lineales inventados por Robert Gallagher en su tesis para doctorado (Gallagher, 1962). Al igual que la mayoría de las tesis, estos códigos pronto fueron olvidados, y no fue sino hasta 1995 que se reinventaron gracias a los avances en el poder de las computadoras, pues ya era práctico utilizarlos.

En un código LDPC, cada bit de salida se forma sólo a partir de una fracción de los bits de entrada. Esto conduce a una representación matricial del código con una densidad baja de 1s, razón por la cual tiene ese nombre. Las palabras codificadas recibidas se decodifican con un algoritmo de aproximación que mejora de manera reiterativa con base en el mejor ajuste de los datos recibidos con una palabra codificada válida. Esto corrige los errores.

Los códigos LDPC son prácticos para tamaños grandes de bloques y tienen excelentes habilidades de corrección de errores que superan a las de muchos otros códigos (incluyendo los que vimos antes) en la práctica. Por esta razón se están incluyendo rápidamente en los nuevos protocolos. Forman parte del estándar para la difusión de video digital, la Ethernet de 10 Gbps, las redes de líneas eléctricas y la versión más reciente de 802.11. Es muy probable que veamos más sobre estos códigos en las futuras redes.

3.2.2 Códigos de detección de errores

Los códigos de corrección de errores se utilizan de manera amplia en los enlaces inalámbricos, que son notoriamente más ruidosos y propensos a errores si se les compara con la fibra óptica. Sin los códigos de corrección de errores sería difícil hacer pasar cualquier cosa. Sin embargo, a través de la fibra óptica o del cable de cobre de alta calidad, la tasa de error es mucho más baja, por lo que la detección de errores y la retransmisión por lo general son más eficientes para manejar un error ocasional.

En esta sección examinaremos tres códigos de detección de errores distintos. Todos son códigos de bloques sistemáticos lineales:

1. Paridad.
2. Sumas de verificación.
3. Pruebas de Redundancia Cíclica (CRC).

Para ver cómo pueden ser más eficientes que los códigos de corrección de errores, considere el primer código de detección de errores en el que se adjunta un solo **bit de paridad** a los datos. El bit de paridad se elige de manera que el número de bits 1 en la palabra codificada sea par (o impar). Hacer esto es equivalente a calcular el bit de paridad (par) como la suma módulo 2 o el resultado de un XOR en los bits de datos. Por ejemplo, cuando se envía la secuencia 1011010 en paridad par, se agrega un bit al final para convertirla en 10110100. Con paridad impar, 1011010 se convierte en 10110101. Un código con un solo bit de paridad tiene una distancia de 2, ya que cualquier error de un solo bit produce una palabra codificada con la paridad incorrecta. Esto significa que puede detectar errores de un solo bit.

Considere un canal en el que los errores son aislados y la tasa de error es de 10^{-6} por bit. Ésta puede parecer una tasa de error pequeña, pero a lo más es una tasa equitativa para un cable de cobre extenso

que desafía a la detección de errores. Los enlaces de LAN comunes proveen tasas de error de bits de 10^{-10}. Sea el tamaño de bloque 1000 bits. Para proporcionar corrección de errores en bloques de 1000 bits, sabemos por la ecuación (3-1) que se requieren 10 bits de verificación. Así, un megabit de datos requeriría 10 000 bits de verificación. Para detectar un solo bloque con 1 bit de error, basta con un bit de paridad por bloque. Por cada 1000 bloques se encontrará un bloque con error y se tendrá que transmitir un bloque extra (1001 bits) para reparar ese error. La sobrecarga total del método de detección de errores y retransmisión es de sólo 2001 bits por megabit de datos, en comparación con los 10 000 bits en un código de Hamming.

Un problema con este esquema es que un bit de paridad sólo puede detectar de manera confiable un error de un solo bit en el bloque. Si el bloque está muy confuso debido a una ráfaga de errores larga, la probabilidad de detectar ese error es sólo de 0.5, lo cual es difícilmente aceptable. Es posible aumentar la probabilidad de manera considerable si cada bloque a enviar se trata como una matriz rectangular de n bits de ancho por k bits de alto. Ahora, si calculamos y enviamos un bit de paridad para cada fila, se detectarán de manera confiable errores de hasta k bits siempre y cuando haya a lo mucho un error por fila.

Pero hay algo más que podemos hacer para ofrecer una mejor protección contra los errores: calcular los bits de paridad sobre los datos en un orden distinto al que se transmiten los bits de datos. A este proceso se le denomina **intercalado**. En este caso, calculamos un bit de paridad para cada una de las n columnas y enviamos todos los bits de datos como k filas; para ello enviamos las filas de arriba hacia abajo y los bits en cada fila de izquierda a derecha de la manera usual. En la última fila enviamos los n bits de paridad. Este orden de transmisión se muestra en la figura 3-8 para $n = 7$ y $k = 7$.

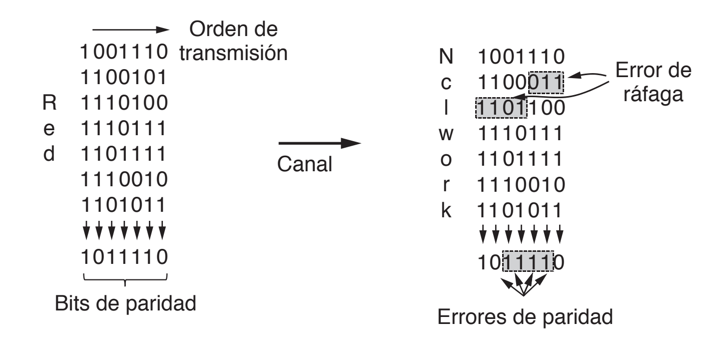


Figura 3-8. Intercalado de bits de paridad para detectar un error de ráfaga.

El intercalado es una técnica general para convertir un código que detecta (o corrige) los errores aislados en uno que detecta (o corrige) los errores de ráfaga. Cuando en la figura 3-8 ocurre un error de ráfaga de longitud $n = 7$, los bits con error se esparcen a través de distintas columnas (un error de ráfaga no implica que todos esos bits estén mal; sólo implica que por lo menos el primero y el último están mal. En la figura 3-8, se voltearon 4 bits en un rango de 7 bits). Por lo menos 1 bit en cada una de las n columnas se verá afectado, de modo que los bits de paridad en esas columnas detectarán el error. Este método usa n bits de paridad en bloques de kn bits de datos para detectar un solo error de ráfaga de longitud n o menor.

Sin embargo, una ráfaga de longitud $n + 1$ pasará sin ser detectada si el primer bit está invertido, el último bit está invertido y todos los demás bits son correctos. Si el bloque está muy alterado por una ráfaga continua o por múltiples ráfagas más cortas, la probabilidad de que cualquiera de las n columnas tenga por accidente la paridad correcta es de 0.5, por lo que la probabilidad de aceptar un bloque alterado cuando no se debe es de 2^{-n}.

El segundo tipo de código de detección de errores, la **suma de verificación**, está muy relacionado con los grupos de bits de paridad. La palabra "suma de verificación" se utiliza con frecuencia para indicar un grupo de bits de verificación asociados con un mensaje, sin importar cómo se calculen. Un grupo de bits de paridad es un ejemplo de una suma de verificación. Sin embargo, hay otras sumas de verificación más poderosas basadas en la suma acumulada de los bits de datos del mensaje. Por lo general la suma de verificación se coloca al final del mensaje, como el complemento de la función de suma. De esta forma, los errores se pueden detectar al sumar toda la palabra codificada recibida, tanto los bits de datos como la suma de verificación. Si el resultado es cero, no se ha detectado ningún error.

Un ejemplo es la suma de verificación de Internet de 16 bits que se utiliza en todos los paquetes de Internet como parte del protocolo IP (Braden y colaboradores, 1988). Esta suma de verificación es una suma de los bits del mensaje divididos en palabras de 16 bits. Como este método opera sobre palabras en vez de bits, como en la paridad, los errores que no modifican la paridad de todas formas pueden alterar la suma y ser detectados. Por ejemplo, si el bit de menor orden en dos palabras distintas se voltea de 0 a 1, una verificación de paridad a lo largo de estos bits no podría detectar un error. Sin embargo, se agregarán dos 1s a la suma de verificación de 16 bits para producir un resultado distinto. Entonces se podrá detectar el error.

La suma de verificación de Internet se calcula en aritmética de complemento a uno, en vez de la suma módulo 2^{16}. En aritmética de complemento a uno, un número negativo es el complemento a nivel de bits de su contraparte positiva. La mayoría de las computadoras modernas emplean aritmética de complemento a dos, en donde un número negativo es el complemento a uno más uno. En una computadora con complemento a dos, la suma de complemento a uno es equivalente a obtener la suma módulo 2^{16} y añadir cualquier desbordamiento de los bits de mayor orden de vuelta a los bits de menor orden. Este algoritmo nos da una cobertura más uniforme de los datos mediante los bits de suma de verificación. En caso contrario, se podrían sumar dos bits de mayor orden, desbordarse y perderse sin cambiar la suma. También hay otro beneficio. El complemento a uno tiene dos representaciones de cero, todos los bits en 0 y todos los bits en 1. Esto permite un valor (por ejemplo, todos los bits en 0) para indicar que no hay suma de verificación sin necesidad de otro campo.

Durante décadas siempre se ha hecho la suposición de que las tramas en las que se empleará la suma de verificación contienen bits aleatorios. Todos los análisis de los algoritmos de suma de verificación se han realizado con base en esta suposición. La inspección de datos reales por parte de Partridge y colaboradores (1995) ha demostrado que esta suposición es bastante incorrecta. Como consecuencia, los errores no detectados son en algunos casos más comunes de lo que se había pensado antes.

En especial, la suma de verificación de Internet es eficiente y simple, pero ofrece una protección débil en algunos casos, precisamente debido a que es una suma simple. No detecta la eliminación o adición de datos cero, ni el intercambio de partes del mensaje y ofrece una protección débil contra los empalmes de mensajes, en donde se juntan partes de dos paquetes. Estos errores pueden parecer muy poco probables de ocurrir mediante procesos aleatorios, pero son justo el tipo de errores que pueden ocurrir con hardware defectuoso.

Hay una mejor opción: la **suma de verificación de Fletcher** (Fletcher, 1982). Ésta incluye un componente posicional, en donde se suma el producto de los datos y su posición con la suma acumulada. Este método ofrece una detección más poderosa de los cambios en la posición de los datos.

Aunque los dos esquemas anteriores pueden ser adecuados algunas veces en capas superiores, en la práctica se utiliza mucho un tercer tipo de código de detección de errores más potente en la capa de enlace: el **CRC** (**Comprobación de Redundancia Cíclica**, del inglés *Cyclic Redundancy Check*), también conocido como **código polinomial**. Los códigos polinomiales se basan en el tratamiento de cadenas de bits como representaciones de polinomios con coeficientes de 0 y 1 solamente. Una trama de k bits se considera como la lista de coeficientes de un polinomio con k términos que van de x^{k-1} a x^0. Se dice que tal polinomio es de grado $k - 1$. El bit de orden mayor (que se encuentra más a la izquierda) es el coeficiente

de x^{k-1}, el siguiente bit es el coeficiente de x^{k-2} y así sucesivamente. Por ejemplo, 110001 tiene 6 bits y, por lo tanto, representa un polinomio de seis términos con coeficientes 1, 1, 0, 0, 0 y 1: $1x^5 + 1x^4 + 0x^3 + 0x^2 + 0x^1 + 1x^0$.

La aritmética polinomial se hace mediante una operación módulo 2, de acuerdo con las reglas de la teoría de campos algebraicos. No hay acarreos para la suma, ni préstamos para la resta. Tanto la suma como la resta son idénticas a un OR exclusivo. Por ejemplo:

10011011	00110011	11110000	01010101
+ 11001010	+ 11001101	− 10100110	− 10101111
01010001	11111110	01010110	11111010

La división larga se lleva a cabo de la misma manera que en binario, excepto que la resta es módulo 2, igual que antes. Se dice que un divisor "cabe" en un dividendo si éste tiene tantos bits como el divisor.

Cuando se emplea el método de código polinomial, el emisor y el receptor deben acordar por adelantado un **polinomio generador**, $G(x)$. Tanto los bits de orden mayor y menor del generador deben ser 1. Para calcular el CRC para una trama con m bits, correspondiente al polinomio $M(x)$, la trama debe ser más larga que el polinomio generador. La idea es incluir un CRC al final de la trama de tal manera que el polinomio representado por la trama con suma de verificación sea divisible entre $G(x)$. Cuando el receptor recibe la trama con la suma de verificación, intenta dividirla entre $G(x)$. Si hay un residuo, ha ocurrido un error de transmisión.

El algoritmo para calcular el CRC es el siguiente:

1. Sea r el grado de $G(x)$. Anexe r bits cero al final de la trama para que ahora contenga $m + r$ bits y corresponda al polinomio $x^rM(x)$.
2. Divida la cadena de bits correspondiente a $G(x)$ entre la correspondiente a $x^rM(x)$; usando una división módulo 2.
3. Reste el residuo (que siempre es de r o menos bits) a la cadena de bits correspondiente a $x^rM(x)$; usando una resta módulo 2. El resultado es la trama con suma de verificación que va a transmitirse. Llame a su polinomio $T(x)$.

La figura 3-9 ilustra el cálculo para una trama 1101011111 utilizando el generador $G(x) = x^4 + x + 1$.

Debe quedar claro que $T(x)$ es divisible (módulo 2) entre $G(x)$. En cualquier problema de división, si se resta el residuo del dividendo, lo que queda es divisible entre el divisor. Por ejemplo, en base 10, si se divide 210 278 entre 10 941, el residuo es 2 399. Si se resta 2 399 a 210 278, lo que queda (207 879) es divisible entre 10 941.

Ahora analizaremos el alcance de este método. ¿Qué tipos de errores se detectarán? Imagine que ocurre un error de transmisión tal que en lugar de que llegue la cadena de bits para $T(x)$, llega $T(x) + E(x)$. Cada bit 1 en $E(x)$ corresponde a un bit que ha sido invertido. Si hay k bits 1 en $E(x)$, han ocurrido k errores de un solo bit. Una ráfaga de errores individual se caracteriza por un 1 inicial, una mezcla de ceros y unos, y un 1 final, siendo los demás bits 0.

Al recibir la trama con suma de verificación, el receptor la divide entre $G(x)$; es decir, calcula $[T(x) + E(x)]/G(x)$. $T(x)/G(x)$ es 0, por lo que el resultado del cálculo es simplemente $E(x)/G(x)$. No se detectarán los errores que por casualidad correspondan a polinomios que contengan $G(x)$ como factor; todos los demás errores serán detectados.

Si ha ocurrido un error de un solo bit, $E(x) = x^i$, donde i determina qué bit es erróneo. Si $G(x)$ contiene dos o más términos, nunca será divisor exacto de $E(x)$, por lo que se detectarán los errores de un solo bit.

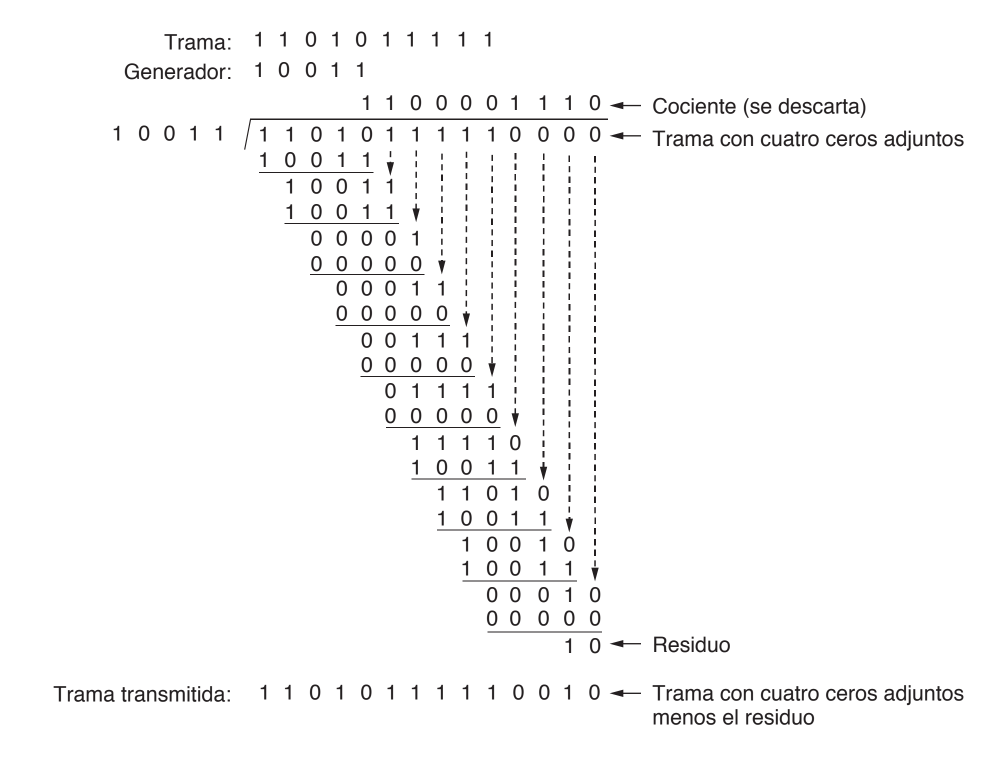


Figura 3-9. Ejemplo del cálculo de CRC.

Si han ocurrido dos errores de un solo bit aislados, $E(x) = x^i + x^j$, donde $i > j$. Esto también se puede escribir como $E(x) = x^j(x^{i-j} + 1)$. Si suponemos que $G(x)$ no es divisible entre x, una condición suficiente para detectar todos los errores dobles es que $G(x)$ no divida a $x^k + 1$ para ninguna k hasta el valor máximo de $i - j$ (es decir, hasta la longitud máxima de la trama). Se conocen polinomios sencillos de bajo grado que dan protección a tramas largas. Por ejemplo, $x^{15} + x^{14} + 1$ no será divisor exacto de $x^k + 1$ para ningún valor de k menor que 32 768.

Si hay una cantidad impar de bits con error, $E(x)$ contiene un número impar de términos (por ejemplo, $x^5 + x^2 + 1$, pero no $x^2 + 1$). Curiosamente, ningún polinomio con un número impar de términos posee a $x + 1$ como un factor en el sistema de módulo 2. Al hacer de $x + 1$ un factor de $G(x)$, podemos detectar todos los errores con un número impar de bits invertidos.

Por último, y lo que es más importante, un código polinomial con r bits de verificación detectará todos los errores en ráfaga de longitud $\leq r$. Un error en ráfaga de longitud k se puede representar mediante $x^i(x^{k-1} + \ldots + 1)$, donde i determina la distancia a la que se encuentra la ráfaga desde el extremo derecho de la trama recibida. Si $G(x)$ contiene un término x^0, no tendrá a x^i como factor, por lo que, si el grado de la expresión entre paréntesis es menor que el grado de $G(x)$, el residuo nunca podrá ser cero.

Si la longitud de la ráfaga es de $r + 1$, el residuo de la división entre $G(x)$ será cero si y sólo si la ráfaga es idéntica a $G(x)$. Según la definición de ráfaga, el primero y el último bit deben ser 1, así que el que sean iguales o no depende de los $r - 1$ bits intermedios. Si se consideran igualmente probables todas las combinaciones, la probabilidad de que se acepte como válida tal trama incorrecta es de $1/2^{r-1}$.

También podemos demostrar que cuando ocurre una ráfaga de errores mayor que $r + 1$ bits, o cuando ocurren varias ráfagas más cortas, la probabilidad de que una trama incorrecta no sea detectada es de $1/2^r$, suponiendo que todos los patrones de bits tengan la misma probabilidad.

Ciertos polinomios se han vuelto estándares internacionales. El que se utiliza en el IEEE 802 se basa en el ejemplo de Ethernet y es:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x^{1} + 1$$

Entre otras propiedades deseables, detecta todas las ráfagas con una longitud de 32 o menor y todas las ráfagas que afecten a un número impar de bits. Se ha utilizado en muchas partes desde la década de 1980. Sin embargo, esto no significa que sea la mejor opción. Mediante el uso de una búsqueda computacional exhaustiva, Castagnoli y colaboradores (1993), junto con Koopman (2002), encontraron los mejores CRC. Estos CRC tienen una distancia de Hamming de 6 para los tamaños de mensajes típicos, mientras que el estándar de IEEE CRC-32 tiene una distancia de Hamming de sólo 4.

Aunque el cálculo requerido para obtener el CRC puede parecer complicado, es fácil calcular y verificar CRC en el hardware mediante circuitos simples de registros de desplazamiento (Peterson y Brown, 1961). En la práctica, casi siempre se usa este hardware. La mayoría de los estándares de red incluyen varios CRC, entre los cuales están casi todas las redes LAN (por ejemplo, Ethernet, 802.11) y los enlaces punto a punto (por ejemplo, paquetes a través de SONET).

3.3 PROTOCOLOS ELEMENTALES DE ENLACE DE DATOS

Para introducir el tema de los protocolos, comenzaremos por estudiar tres protocolos de complejidad creciente. Los lectores interesados pueden conseguir un simulador de estos protocolos y otros subsecuentes a través de la web (vea el prefacio). Antes de estudiar los protocolos, es útil hacer explícitos algunos de los supuestos implícitos en el modelo de comunicación.

Para comenzar, supongamos que las capas física, de enlace de datos y de red son procesos independientes que se comunican pasando mensajes de un lado a otro. En la figura 3-10 se muestra una implementación común. El proceso de la capa física y una parte del proceso de la capa de enlace de datos se ejecutan en hardware dedicado, conocido como **NIC** (**Tarjeta de Interfaz de Red**, del inglés *Network Interface Card*). El resto del proceso de la capa de enlace y el proceso de la capa de red se ejecutan en la CPU principal como parte del sistema operativo, en donde el software para el proceso de la capa de enlace a menudo toma la forma de un **controlador de dispositivo**. Sin embargo, también puede haber otras implementaciones (por ejemplo, tres procesos descargados a un dispositivo de hardware dedicado, conocido como **acelerador de red**, o tres procesos ejecutándose en la CPU principal en un radio definido por software). En realidad, la implementación preferida cambia de una década a otra con las concesiones tecnológicas. En cualquier caso, el hecho de tratar las tres capas como procesos independientes hace más nítido el análisis en el terreno conceptual y también sirve para enfatizar la independencia de las capas.

Otro supuesto clave es que la máquina *A* desea mandar un flujo considerable de datos a la máquina *B* mediante el uso de un servicio confiable orientado a conexión. Después consideraremos el caso en que *B* también quiere mandar datos a *A* de manera simultánea. Se ha supuesto que *A* tiene un suministro infinito de datos listos para ser enviados y nunca tiene que esperar a que éstos se produzcan, sino que cuando la capa de enlace de datos de *A* los solicita, la capa de red siempre es capaz de proporcionarlos de inmediato (posteriormente también descartaremos esta restricción). Por otro lado, supondremos que las máquinas no fallan. Es decir, estos protocolos manejan errores de comunicación, pero no los problemas causados por computadoras que fallan y se reinician.

En lo que concierne a la capa de enlace de datos, el paquete que recibe a través de la interfaz desde la capa de red sólo es de datos, los cuales deben ser entregados bit por bit a la capa de red del destino.

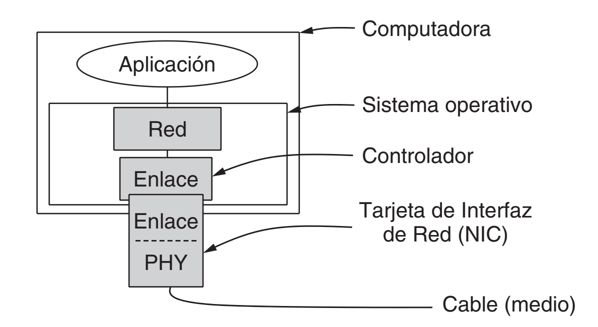


Figura 3-10. Implementación de las capas física, de enlace de datos y de red.

El hecho de que la capa de red del destino pueda interpretar parte del paquete como un encabezado no es de importancia para la capa de enlace de datos.

Cuando la capa de enlace de datos acepta un paquete, lo encapsula en una trama agregándole un encabezado y un terminador de enlace de datos (vea la figura 3-1). Por lo tanto, una trama consiste en un paquete incrustado, con cierta información de control (en el encabezado) y una suma de verificación (en el terminador). A continuación la trama se transmite a la capa de enlace de datos de la otra máquina. Supondremos que existen los procedimientos de biblioteca adecuados *to_physical_layer* para enviar una trama y *from_physical_layer* para recibir una trama. Estos procedimientos calculan y anexan o verifican la suma de verificación (que por lo general se realiza en el hardware) de modo que no tengamos qué preocuparnos por ella como parte de los protocolos que desarrollaremos en esta sección. Podrían usar el algoritmo de CRC que vimos en la sección anterior, por ejemplo.

En un principio, el receptor no tiene nada que hacer. Sólo está esperando a que ocurra algo. En los protocolos de ejemplo de este capítulo indicaremos que la capa de enlace de datos está en espera de que ocurra algo mediante la llamada al procedimiento *wait_ for_event(&event)*. Este procedimiento sólo regresa cuando ocurre algo (por ejemplo, cuando llega una trama). Al regresar, la variable *event* indica lo que ha ocurrido. El conjunto de eventos posibles es distinto para cada uno de los diferentes protocolos que describiremos, y se definirán por separado para cada protocolo. Tenga en cuenta que en una situación más realista, la capa de enlace de datos no se quedará en un ciclo cerrado esperando un evento, como hemos sugerido, sino que recibirá una interrupción, la que ocasionará que suspenda lo que estaba haciendo y proceda a manejar la trama entrante. Sin embargo, por simplicidad ignoraremos todos los detalles de la actividad paralela en la capa de enlace de datos y daremos por hecho que la capa está dedicada de tiempo completo a manejar nuestro único canal.

Cuando llega una trama al receptor, se vuelve a calcular la suma de verificación. Si la suma de verificación en la trama es incorrecta (es decir, si hubo un error de transmisión), se informa esto a la capa de enlace de datos (*event* = *cksum_err*). Si la trama entrante llega sin daño, también se le informa a la capa de enlace de datos (*event* = *frame_arrival*) de modo que pueda adquirir la trama para inspeccionarla mediante el uso de *from_physical_layer*. Tan pronto como la capa de enlace de datos receptora adquiere una trama sin daños, verifica la información de control del encabezado y, si todo está bien, pasa la parte que corresponde al paquete a la capa de red. En ninguna circunstancia se entrega un encabezado de trama a una capa de red.

Hay una buena razón por la que la capa de red nunca debe recibir ninguna parte del encabezado de trama: para mantener completamente separados el protocolo de red y el de enlace de datos. En tanto la capa de red no sepa nada en absoluto sobre el protocolo de enlace de datos ni el formato de la trama, éstos podrán cambiarse sin requerir cambios en el software de la capa de red. Esto ocurre cada vez que se instala una nueva NIC en una computadora. Al proporcionarse una interfaz rígida entre la capa de red y la de

enlace de datos se simplifica en gran medida la tarea de diseño, pues los protocolos de comunicación de las diferentes capas pueden evolucionar en forma independiente.

```
#define MAX_PKT 1024                                   /* determina el tamaño del paquete en bytes */

typedef enum {false, true} boolean;                    /* tipo booleano */
typedef unsigned int seq_nr;                           /* números de secuencia o confirmación */
typedef struct {unsigned char data[MAX_PKT];}packet;   /* definición del paquete */
typedef enum {data, ack, nak} frame_kind;              /* definición de frame_kind */

typedef struct {                                       /* las tramas se transportan en esta capa */
  frame_kind kind;                                     /* ¿qué tipo de trama es? */
  seq_nr seq;                                          /* número de secuencia */
  seq_nr ack;                                          /* número de confirmación de recepción */
  packet info;                                         /* paquete de la capa de red */
} frame;

/* Espera a que ocurra un evento; devuelve el tipo en la variable event. */
void wait_for_event(event_type *event);

/* Obtiene un paquete de la capa de red para transmitirlo por el canal. */
void from_network_layer(packet *p);

/* Entrega información de una trama entrante a la capa de red. */
void to_network_layer(packet *p);

/* Obtiene una trama entrante de la capa física y la copia en r. */
void from_physical_layer(frame *r);

/* Pasa la trama a la capa física para transmitirla. */
void to_physical_layer(frame *s);

/* Arranca el reloj y habilita el evento de expiración de temporizador. */
void start_timer(seq_nr k);

/* Detiene el reloj y deshabilita el evento de expiración de temporizador. */
void stop_timer(seq_nr k);

/* Inicia un temporizador auxiliar y habilita el evento ack_timeout. */
void start_ack_timer(void);

/* Detiene el temporizador auxiliar y deshabilita el evento ack_timeout. */
void stop_ack_timer(void);

/* Permite que la capa de red cause un evento network_layer_ready. */
void enable_network_layer(void);

/* Evita que la capa de red cause un evento network_layer_ready. */
void disable_network_layer(void);

/* La macro inc se expande en línea: incrementa circularmente a k. */
#define inc(k) if (k < MAX_SEQ) k = k + 1; else k = 0
```

Figura 3-11. Algunas definiciones necesarias en los protocolos que siguen. Estas definiciones se encuentran en el archivo *protocol.h.*

En la figura 3-11 se muestran algunas declaraciones comunes (en C) para muchos de los protocolos que analizaremos después. Allí se definen cinco estructuras de datos: *boolean*, *seq_nr*, *packet*, *frame_kind*

y *frame*. Un *boolean* es un tipo enumerado que puede tener los valores *true* y *false*. Un *seq_nr* (número de secuencia) es un entero pequeño que sirve para numerar las tramas, con el fin de distinguirlas. Estos números de secuencia van de 0 hasta *MAX_SEQ* (inclusive), el cual es definido en cada protocolo según sus necesidades. Un *packet* es la unidad de intercambio de información entre la capa de red y la de enlace de datos en la misma máquina, o entre entidades iguales de la capa de red. En nuestro modelo siempre contiene *MAX_PKT* bytes, pero en la práctica sería de longitud variable.

Un *frame* está compuesto de cuatro campos: *kind*, *seq*, *ack* e *info*. Los primeros tres contienen información de control y el último puede contener los datos por transferir. Estos campos de control constituyen en conjunto el **encabezado de la trama**.

El campo *kind* indica si hay datos en la trama, ya que algunos de los protocolos distinguen entre las tramas que contienen sólo información de control y las que también contienen datos. Los campos *seq* y *ack* se emplean para números de secuencia y confirmaciones de recepción, respectivamente; describiremos su uso más adelante con mayor detalle. El campo *info* de una trama de datos contiene un solo paquete; el campo *info* de una trama de control no se usa. En una implementación más realista se usaría un campo *info* de longitud variable, el cual se omitiría por completo en las tramas de control.

Es importante entender la relación entre un paquete y una trama. Para construir un paquete, la capa de red toma un mensaje de la capa de transporte y le agrega el encabezado de la capa de red. Después este paquete se pasa a la capa de enlace de datos para incluirlo en el campo *info* de una trama saliente. Cuando ésta llega a su destino, la capa de enlace de datos extrae de ella el paquete y a continuación lo pasa a la capa de red. De esta manera, la capa de red puede actuar como si las máquinas pudieran intercambiar paquetes directamente.

En la figura 3-11 también se listan varios procedimientos, los cuales son rutinas de biblioteca cuyos detalles dependen de la implementación, por lo que no nos ocuparemos de su funcionamiento interno. El procedimiento *wait_ for_event* se queda en un ciclo cerrado esperando que algo ocurra, como se mencionó antes. La capa de enlace de datos usa los procedimientos *to_network_layer* y *from_network_layer* para pasar paquetes a la capa de red y aceptar paquetes de ella, respectivamente. Cabe mencionar que *from_physical_layer* y *to_physical_layer* pasan tramas entre la capa de enlace de datos y la capa física. En otras palabras, *to_network_layer* y *from_network_layer* tienen que ver con la interfaz entre las capas 2 y 3, mientras que *from_physical_layer* y *to_physical_layer* tratan con la interfaz entre las capas 1 y 2.

En la mayoría de los protocolos suponemos que el canal es no confiable y en ocasiones pierde tramas completas. Para poder recuperarse de tales calamidades, la capa de enlace de datos emisora debe iniciar un temporizador o reloj interno cada vez que envía una trama. Si no obtiene respuesta después de cierto intervalo predeterminado, el reloj expira y la capa de enlace de datos recibe una señal de interrupción.

En nuestros protocolos, para manejar esto es necesario permitir que el procedimiento *wait_for_event* devuelva *event = timeout*. Los procedimientos *start_timer* y *stop_timer* inician y detienen el temporizador, respectivamente. Los eventos de expiración del temporizador sólo son posibles cuando éste se encuentra en funcionamiento y antes de llamar a *stop_timer*. Se permite de manera explícita llamar a *start_timer* cuando el temporizador está en operación; tal llamada tan sólo restablece el reloj para hacer que el temporizador termine después de haber transcurrido un intervalo completo de temporización (a menos que se restablezca o apague antes).

Los procedimientos *start_ack_timer* y *stop_ack_timer* controlan un temporizador auxiliar que se utiliza para generar confirmaciones de recepción en ciertas condiciones.

Los procedimientos *enable_network_layer* y *disable_network_layer* se usan en los protocolos más sofisticados, en los que ya no suponemos que la capa de red siempre tiene paquetes para enviar. Cuando

la capa de enlace de datos habilita a la capa de red, ésta tiene permitido interrumpir cada vez que tenga que enviar un paquete. Esto lo indicamos con *event = network_layer_ready*. Cuando la capa de red está deshabilitada, no puede causar dichos eventos. Si tiene cuidado respecto a cuándo debe habilitar y deshabilitar su capa de red, la capa de enlace de datos puede evitar que la capa de red la sature con paquetes para los que no tiene espacio de búfer.

Los números de secuencia de las tramas siempre están en el intervalo de 0 a *MAX_SEQ* (inclusive), en donde *MAX_SEQ* es diferente para los distintos protocolos. Con frecuencia es necesario avanzar circularmente en 1 un número de secuencia (por ejemplo, *MAX_SEQ* va seguido de 0). La macro *inc* lleva a cabo este incremento. Esta función se ha definido como macro porque se usa en línea dentro de la ruta crítica. Como veremos después, el factor que limita con frecuencia el desempeño de una red es el procesamiento del protocolo, por lo que definir como macros las operaciones sencillas como ésta no afecta la legibilidad del código y sí mejora el desempeño.

Las declaraciones de la figura 3-11 son parte de todos los protocolos que estudiaremos en breve. Para ahorrar espacio y proveer una referencia conveniente, se han extraído y listado juntas, pero conceptualmente deberían estar integradas con los protocolos mismos. En C, para llevar a cabo esta integración se colocan las definiciones en un archivo especial de encabezado, en este caso *protocol.h*, y se utiliza la directiva *#include* del preprocesador de C para incluirlas en los archivos de protocolos.

3.3.1 Un protocolo simplex utópico

Como ejemplo inicial consideraremos un protocolo que es lo más sencillo posible, por la posibilidad de que algo salga mal. Los datos son transmitidos en una sola dirección; las capas de red tanto del emisor como del receptor siempre están listas. Podemos ignorar el tiempo de procesamiento. Hay un espacio infinito de búfer disponible. Y lo mejor de todo, el canal de comunicación entre las capas de enlace de datos nunca daña ni pierde las tramas. Este protocolo completamente irreal, al que apodaremos “utopía”, es simplemente para mostrar la estructura básica en la que nos basaremos. Su implementación se muestra en la figura 3-12.

El protocolo consiste en dos procedimientos diferentes, un emisor y un receptor. El emisor se ejecuta en la capa de enlace de datos de la máquina de origen y el receptor se ejecuta en la capa de enlace de datos de la máquina de destino. No se usan números de secuencia ni confirmaciones de recepción, por lo que no se necesita *MAX_SEQ*. El único tipo de evento posible es *frame_arrival* (es decir, la llegada de una trama sin daños).

El emisor está en un ciclo **while** infinito que sólo envía datos a la línea tan rápido como puede. El cuerpo del ciclo consiste en tres acciones: obtener un paquete de la (siempre dispuesta) capa de red, construir una trama de salida usando la variable *s* y enviar la trama a su destino. Este protocolo sólo utiliza el campo *info* de la trama, pues los demás campos tienen que ver con el control de errores y de flujo, y aquí no hay restricciones de este tipo.

El receptor también es sencillo. Al principio espera que algo ocurra, siendo la única posibilidad la llegada de una trama sin daños. En algún momento llega la trama, el procedimiento *wait_for_event* regresa y *event* contiene el valor *frame_arrival* (que de todos modos se ignora). La llamada a *from_physical_layer* elimina la trama recién llegada del búfer de hardware y la coloca en la variable *r*, en donde el código receptor pueda obtenerla. Por último, la parte de los datos se pasa a la capa de red y la capa de enlace de datos se retira para esperar la siguiente trama, para lo cual se suspende efectivamente hasta que ésta llega.

El protocolo utópico es irreal, ya que no maneja el control de flujo ni la corrección de errores. Su procesamiento se asemeja al de un servicio sin conexión ni confirmación de recepción que depende de las capas más altas para resolver estos problemas, aun cuando un servicio sin conexión ni confirmación de recepción realizaría cierta detección de errores.

```
/* El protocolo 1 (utopía) provee la transmisión de datos en una sola dirección, del emisor al receptor. Se supone
   que el canal de comunicación está libre de errores, y que el receptor es capaz de procesar todas las entra-
   das a una rapidez infinita. En consecuencia, el emisor se mantiene en un ciclo, enviando datos a la línea tan
   rapido como puede. */

typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender1(void)
{
 frame s;                                     /* búfer para una trama de salida */
 packet buffer;                               /* búfer para un paquete de salida */

 while (true) {
     from_network_layer(&buffer);             /* consigue algo qué enviar */
     s.info = buffer;                         /* lo copia en s para transmitirlo */
     to_physical_layer(&s);                   /* lo envía a su destino */
 }                                            /* Mañana, y mañana, y mañana,
                                              Se arrastra a este mísero paso de día a día
                                              Hasta la última sílaba del tiempo recordado
                                              – Macbeth, V, v */
}

void receiver1(void)
{
 frame r;
 event_type event;                            /* ocupado por wait, pero no se usa aquí */

 while (true) {
     wait_for_event(&event);                  /* la única posibilidad es frame_arrival */
     from_physical_layer(&r);                 /* obtiene la trama entrante */
     to_network_layer(&r.info);               /* pasa los datos a la capa de red */
 }
}
```

Figura 3-12. Un protocolo simplex utópico.

3.3.2 Protocolo simplex de parada y espera para un canal libre de errores

Ahora debemos lidiar con el problema principal de evitar que el emisor sature al receptor enviando tramas a una mayor velocidad de la que este último puede procesarlas. Esta situación puede ocurrir con facilidad en la práctica, por lo que es de extrema importancia evitarla. Sin embargo, aún existe el supuesto de que el canal está libre de errores y el tráfico de datos sigue siendo simplex.

Una solución es construir un receptor lo suficientemente poderoso como para procesar un flujo continuo de tramas, una tras otra sin interrupción (lo equivalente sería definir la capa de enlace de modo que fuera lo bastante lenta como para que el receptor pudiera mantenerse a la par). Debe tener suficiente capacidad en el búfer y de procesamiento como para operar a la tasa de transmisión de la línea; asimismo debe ser capaz de pasar las tramas que se reciben en la capa de red con la rapidez suficiente. Sin embargo, ésta es una solución para el peor de los casos. Requiere hardware dedicado y se pueden desperdiciar recursos si el enlace se usa poco. Además, sólo cambia el problema de tratar con un emisor demasiado rápido a otra parte; en este caso, a la capa de red.

Una solución más general para este dilema es hacer que el receptor proporcione retroalimentación al emisor. Tras haber pasado un paquete a su capa de red, el receptor regresa al emisor una pequeña trama ficticia que, de hecho, autoriza al emisor para que transmita la siguiente trama. Después de enviar una

trama, el protocolo exige que el emisor espere hasta que llegue la pequeña trama ficticia (es decir, la confirmación de recepción). Este retraso es un ejemplo simple de un protocolo de control de flujo.

Los protocolos en los que el emisor envía una trama y luego espera una confirmación de recepción antes de continuar se denominan de **parada y espera**. En la figura 3-13 se da un ejemplo de un protocolo simplex de parada y espera.

Aunque el tráfico de datos en este ejemplo es simplex, y va sólo desde el emisor al receptor, las tramas viajan en ambas direcciones. En consecuencia, el canal de comunicación entre las dos capas de enlace de datos necesita tener capacidad de transferencia de información bidireccional. Sin embargo, este protocolo implica una alternancia estricta de flujo: primero el emisor envía una trama, después el receptor envía una trama, después el emisor envía otra trama, luego el receptor envía otra, y así sucesivamente. Aquí sería suficiente un canal físico semi-dúplex.

Al igual que en el protocolo 1, el emisor comienza obteniendo un paquete de la capa de red, lo usa para construir una trama y enviarla a su destino. Sólo que ahora, a diferencia del protocolo 1, el emisor debe esperar hasta que llegue una trama de confirmación de recepción antes de reiniciar el ciclo y obtener el siguiente paquete de la capa de red. La capa de enlace de datos emisora ni siquiera necesita inspeccionar la trama entrante, ya que sólo hay una posibilidad. La trama entrante siempre es de confirmación de recepción.

```
/* El protocolo 2 (parada y espera) también provee un flujo unidireccional de datos del emisor al receptor. Se
   da por hecho nuevamente que el canal de comunicación está libre de errores, como en el protocolo 1. Sin
   embargo, esta vez el receptor tiene capacidad finita de búfer y capacidad finita de procesamiento, por lo que
   el protocolo debe evitar de manera explícita que el emisor sature al receptor con datos a una mayor velocidad
   de la que pueda manejar. */

typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender2(void)
{
  frame s;                                   /* búfer para una trama de salida */
  packet buffer;                             /* búfer para un paquete de salida */
  event_type event;                          /* frame_arrival es la única posibilidad */

while (true) {
      from_network_layer(&buffer);           /* consigue algo que enviar */
      s.info = buffer;                       /* lo copia en s para transmitirlo */
   to_physical_layer(&s);                    /* adiós a la pequeña trama */
   wait_for_event(&event);                   /* no procede hasta que recibe la señal de continuación */
  }
}

void receiver2(void)
{
  frame r, s;                                /* búferes para las tramas */
  event_type event;                          /* frame_arrival es la única posibilidad */
  while (true) {
      wait_for_event(&event);                /* la única posibilidad es frame_arrival */
      from_physical_layer(&r);               /* obtiene la trama entrante */
      to_network_layer(&r.info);             /* pasa los datos a la capa de red */
      to_physical_layer(&s);                 /* envía una trama ficticia para despertar al emisor */
  }
}
```

Figura 3-13. Un protocolo simplex de parada y espera.

La única diferencia entre *receiver1* y *receiver2* es que, tras entregar un paquete a la capa de red, *receiver2* regresa al emisor una trama de confirmación de recepción antes de entrar nuevamente en el ciclo de espera. Puesto que sólo es importante la llegada de la trama al emisor y no su contenido, el receptor no necesita poner ninguna información específica en la trama.

3.3.3 Protocolo simplex de parada y espera para un canal ruidoso

Ahora consideremos la situación normal de un canal de comunicación que comete errores. Las tramas pueden llegar dañadas o se pueden perder por completo. Sin embargo, suponemos que si una trama se daña en tránsito, el hardware del receptor detectará esto cuando calcule la suma de verificación. Si la trama está dañada de tal manera que pese a ello la suma de verificación sea correcta (una ocurrencia muy poco probable), este protocolo (y todos los demás) puede fallar (es decir, tal vez entregue un paquete incorrecto a la capa de red).

A primera vista puede parecer que funcionaría una variación del protocolo 2: agregar un temporizador. El emisor podría enviar una trama, pero el receptor sólo enviaría una trama de confirmación de recepción si los datos llegaran correctamente. Si llegara una trama dañada al receptor, se desecharía. Después de un tiempo el temporizador del emisor expiraría y éste enviaría la trama otra vez. Este proceso se repetiría hasta que la trama por fin llegara intacta.

Pero el esquema anterior tiene un defecto fatal. Piense en el problema y trate de descubrir lo que podría salir mal antes de continuar leyendo.

Para ver lo que puede resultar mal, recuerde que el objetivo de la capa de enlace de datos es proporcionar una comunicación transparente y libre de errores entre los procesos de las capas de red. La capa de red de la máquina *A* pasa una serie de paquetes a su capa de enlace de datos, la cual debe asegurar que se entregue una serie de paquetes idénticos a la capa de red de la máquina *B* a través de su capa de enlace de datos. En particular, la capa de red en *B* no tiene manera de saber si el paquete se perdió o duplicó, por lo que la capa de enlace de datos debe garantizar que ninguna combinación de errores de transmisión, por improbables que sean, pudiera causar la entrega de un paquete duplicado a la capa de red.

Considere el siguiente escenario:

1. La capa de red de *A* entrega el paquete 1 a su capa de enlace de datos. La máquina *B* recibe correctamente el paquete y lo pasa a su capa de red. *B* regresa a *A* una trama de confirmación de recepción.
2. La trama de confirmación de recepción se pierde por completo. Nunca llega. La vida sería mucho más sencilla si el canal sólo alterara o perdiera tramas de datos y no tramas de control, pero desgraciadamente el canal no hace distinciones.
3. El temporizador de la capa de enlace de datos de A expira en algún momento. Al no haber recibido una confirmación de recepción, supone (incorrectamente) que su trama de datos se ha perdido o dañado, y envía otra vez la trama que contiene el paquete 1.
4. La trama duplicada también llega bien a la capa de enlace de datos de *B* y de ahí se pasa de manera inadvertida a la capa de red. Si *A* está enviando un archivo a *B*, parte del archivo se duplicará (es decir, la copia del archivo reconstruida por B será incorrecta y el error no se habrá detectado). En otras palabras, el protocolo fallará.

Sin duda, lo que se necesita es alguna manera de que el receptor sea capaz de distinguir entre una trama que está viendo por primera vez y una retransmisión. La forma evidente de lograr esto es hacer que el emisor ponga un número de secuencia en el encabezado de cada trama que envía. A continuación, el receptor puede verificar el número de secuencia de cada trama que llega para ver si es una trama nueva o un duplicado que debe descartarse.

Como el protocolo debe ser correcto y es probable que el campo de número de secuencia en el encabezado sea pequeño como para poder usar el enlace en forma eficiente, surge la pregunta: ¿cuál es la cantidad mínima de bits necesarios para el número de secuencia? El encabezado podría proveer 1 bit, unos cuantos bits, 1 byte o varios bytes para un número de secuencia, dependiendo del protocolo. El punto importante es que debe transportar números de secuencia que sean lo bastante grandes como para que el protocolo funcione de manera correcta, o de lo contrario no se podrá considerar un verdadero protocolo.

La única ambigüedad de este protocolo es entre una trama m y su sucesor directo, $m + 1$. Si la trama m se pierde o se daña, el receptor no confirmará su recepción y el emisor seguirá tratando de enviarla. Una vez que la trama se reciba correctamente, el receptor regresará una confirmación de recepción al emisor. Es aquí donde surge el problema potencial. Dependiendo de si el emisor recibe correctamente la trama de confirmación de recepción o no, tratará de enviar m o $m + 1$.

En el emisor, el evento que desencadena la transmisión de la trama $m + 1$ es la llegada de una confirmación de recepción de la trama m. Pero esto implica que $m - 1$ se recibió correctamente; también implica que el emisor recibió correctamente su confirmación de recepción. De otra manera, el emisor no habría comenzado con m y mucho menos hubiera considerado $m + 1$. Como consecuencia, la única ambigüedad es entre una trama y su antecesor o sucesor inmediato, no entre el antecesor y el sucesor.

Por lo tanto, basta con un número de secuencia de 1 bit (0 o 1). En cada instante, el receptor espera un número de secuencia en particular. Cuando llega una trama que contiene el número de secuencia correcto, se acepta y se pasa a la capa de red, para después confirmar su recepción. Luego, el número de secuencia esperado se incrementa módulo 2 (es decir, 0 se vuelve 1 y 1 se vuelve 0). Cualquier trama que llegue y que contenga el número de secuencia incorrecto se rechaza como duplicada. Sin embargo, la última confirmación de recepción válida se repite de modo que el emisor pueda descubrir en un momento dado que se recibió la trama.

En la figura 3-14 se muestra un ejemplo de este tipo de protocolo. Los protocolos en los que el emisor espera una confirmación de recepción positiva antes de avanzar al siguiente elemento de datos se conocen comúnmente como **ARQ** (**Solicitud Automática de Repetición**, del inglés *Automatic Repeat reQuest*) o **PAR** (**Confirmación de Recepción Positiva con Retransmisión**, del inglés *Positive Acknowledgement with Retransmission*). Al igual que el protocolo 2, éste también transmite datos en una sola dirección.

El protocolo 3 difiere de sus antecesores en cuando a que tanto el emisor como el receptor tienen una variable cuyo valor se recuerda mientras la capa de enlace de datos está en estado de espera. El emisor recuerda el número de secuencia de la siguiente trama a enviar en *next_frame_to_send*; el receptor recuerda el número de secuencia de la siguiente trama esperada en *frame_expected*. Cada protocolo tiene una pequeña fase de inicialización antes de entrar en el ciclo infinito.

Después de transmitir una trama, el emisor inicia el temporizador. Si ya estaba en operación, se restablece para conceder otro intervalo completo de temporización. Hay que elegir dicho intervalo de modo que haya suficiente tiempo para que la trama llegue al receptor, éste la procese en el peor caso y la confirmación de recepción se propague de vuelta al emisor. Sólo hasta que haya transcurrido ese intervalo podremos suponer con seguridad que se ha perdido la trama transmitida o su confirmación de recepción, y se debe enviar un duplicado. Si el intervalo establecido es muy pequeño, el emisor transmitirá tramas innecesarias. Si bien estas tramas adicionales no afectarán el funcionamiento correcto del protocolo, sí afectarán el desempeño.

Después de transmitir una trama e iniciar el temporizador, el emisor espera a que ocurra algo interesante. Sólo hay tres posibilidades: que una trama de confirmación de recepción llegue sin daño, que llegue una trama de confirmación de recepción dañada o que expire el temporizador. Si entra una confirmación de recepción válida, el emisor obtiene el siguiente paquete de su capa de red y lo coloca en el búfer, sobreescribiendo el paquete anterior. También incrementa el número de secuencia. Si llega una trama

```
/* El protocolo 3 (PAR) permite el flujo unidireccional de datos por un canal no confiable. */

#define MAX_SEQ 1                                   /* debe ser 1 para el protocolo 3 */
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"

void sender3(void)
{
seq_nr next_frame_to_send;                          /* número de secuencia de la siguiente trama de salida */
frame s;                                            /* variable de trabajo */
packet buffer;                                      /* búfer para un paquete de salida */
event_type event;

next_frame_to_send = 0;                             /* inicializa números de secuencia de salida */
from_network_layer(&buffer);                        /* obtiene el primer paquete */
while (true){
      s.info = buffer;                              /* construye una trama para transmisión */
      s.seq = next_frame_to_send;                   /* inserta un número de secuencia en la trama */
      to_physical_layer(&s);                        /* la envía a su destino */
      start_timer(s.seq);                           /* si la respuesta tarda mucho, expira el temporizador */
      wait_for_event(&event);                       /* frame_arrival, cksum_err, timeout */
      if (event == frame_arrival){
          from_physical_layer(&s);                  /* obtiene la confirmación de recepción */
          if (s.ack == next_frame_to_send){
               stop_timer(s.ack);                   /* desactiva el temporizador */
               from_network_layer(&buffer);         /* obtiene siguiente a enviar */
               inc(next_frame_to_send);             /* invierte next_frame_to_send */
          }
      }
 }
}

void receiver3(void)
{
seq_nr frame_expected;
frame r, s;
event_type event;

frame_expected = 0;
while (true){
      wait_for_event(&event);                       /* posibilidades: frame_arrival, cksum_err */
      if (event == frame_arrival){                  /* ha llegado una trama válida. */
          from_physical_layer(&r);                  /* obtiene la trama recién llegada */
          if (r.seq == frame_expected){             /* esto es lo que hemos estado esperando. */
               to_network_layer(&r.info);           /* pasa los datos a la capa de red */
               inc(frame_expected);                 /* para la próxima se espera el otro número de secuencia */
          }
          s.ack = 1 – frame_expected;               /* indica la trama cuya recepción se está confirmando */
          to_physical_layer(&s);                    /* envía confirmación de recepción */
      }
 }
}
```

Figura 3-14. Un protocolo de confirmación de recepción positiva con retransmisión.

dañada o expira el temporizador, no cambia ni el búfer ni el número de secuencia de modo que se pueda enviar un duplicado. En todos los casos se envía a continuación el contenido del búfer (ya sea el siguiente paquete o un duplicado).

Cuando llega una trama válida al receptor, su número de secuencia se verifica para saber si es un duplicado. Si no lo es, se acepta, se pasa a la capa de red y se genera una confirmación de recepción. Los duplicados y las tramas dañadas no se pasan a la capa de red, pero hacen que se confirme la recepción de la última trama que se recibió correctamente para avisar al emisor de modo que avance a la siguiente trama o retransmita la trama dañada.

3.4 PROTOCOLOS DE VENTANA DESLIZANTE

En los protocolos anteriores, las tramas de datos se transmitían en una sola dirección. En la mayoría de las situaciones prácticas existe la necesidad de transmitir datos en ambas direcciones. Una manera de lograr una transmisión de datos full-dúplex es tener dos instancias de uno de los protocolos anteriores, cada uno de los cuales debe usar un enlace separado para el tráfico de datos simplex (en distintas direcciones). A su vez, cada enlace se compone de un canal de "ida" (para los datos) y de un canal de "retorno" (para las confirmaciones de recepción). En ambos casos se desperdicia la capacidad del canal de retorno casi por completo.

Una mejor idea es utilizar el mismo enlace para datos en ambas direcciones. Después de todo, en los protocolos 2 y 3 ya se usaba para transmitir tramas en ambos sentidos, y por lo general el canal de retorno tiene la misma capacidad que el canal de ida. En este modelo, las tramas de datos de *A* a *B* se entremezclan con las tramas de confirmación de recepción de *A* a *B*. Si analizamos el campo *kind* en el encabezado de una trama entrante, el receptor puede saber si la trama es de datos o de confirmación de recepción.

Aunque intercalar datos y tramas de control en el mismo enlace es una gran mejora respecto al uso de dos enlaces físicos separados, es posible realizar otra mejora. Cuando llega una trama de datos, en lugar de enviar de inmediato una trama de control independiente, el receptor se aguanta y espera hasta que la capa de red le pasa el siguiente paquete. La confirmación de recepción se anexa a la trama de datos de salida (mediante el uso del campo *ack* del encabezado de la trama). En efecto, la confirmación de recepción viaja gratuitamente en la siguiente trama de datos de salida. La técnica de retardar temporalmente las confirmaciones de recepción salientes para que puedan viajar en la siguiente trama de datos de salida se conoce como **superposición** (*piggybacking*).

La principal ventaja de usar la superposición en lugar de tener tramas de confirmación de recepción independientes, es un mejor aprovechamiento del ancho de banda disponible del canal. El campo *ack* del encabezado de la trama ocupa sólo unos cuantos bits, mientras que una trama separada requeriría de un encabezado, la confirmación de recepción y una suma de verificación. Además, el envío de menos tramas casi siempre representa una carga de procesamiento más ligera en el receptor. En el siguiente protocolo que vamos a examinar, el campo de superposición sólo ocupa 1 bit en el encabezado de trama. Pocas veces ocupa más de unos cuantos bits.

Sin embargo, la superposición introduce una complicación inexistente en las confirmaciones de recepción independientes. ¿Cuánto tiempo debe esperar la capa de enlace de datos un paquete al cual pueda superponer la confirmación de recepción? Si la capa de enlace de datos espera más tiempo del que tarda en terminar el temporizador del emisor, se volverá a transmitir la trama y se frustrará el propósito de enviar confirmaciones de recepción. Si la capa de enlace de datos fuera un oráculo y pudiera predecir el futuro, sabría cuándo se recibiría el siguiente paquete de la capa de red y podría decidir entre esperarlo o enviar de inmediato una confirmación de recepción independiente, dependiendo del tiempo de espera proyectado. Por supuesto que la capa de enlace de datos no puede predecir el futuro, de modo que debe recurrir a algún esquema particular para el caso, como esperar un número fijo de milisegundos. Si llega rápidamente un nuevo paquete, la confirmación de recepción se superpone a él. De otra manera, si no ha llegado ningún paquete nuevo al final de este periodo, la capa de enlace de datos simplemente envía una trama de confirmación de recepción independiente.

Los siguientes tres protocolos son bidireccionales y pertenecen a una clase llamada protocolos de **ventana deslizante**. Los tres difieren entre ellos en términos de eficiencia, complejidad y requerimientos de búfer, como veremos más adelante. En ellos, al igual que en todos los protocolos de ventana deslizante, cada trama de salida contiene un número de secuencia que va desde 0 hasta algún número máximo. Por lo general este valor máximo es $2^n - 1$, por lo que el número de secuencia encaja perfectamente en un campo de n bits. El protocolo de ventana deslizante de parada y espera utiliza $n = 1$ y restringe los números de secuencia de 0 y 1, pero las versiones más sofisticadas pueden utilizar un n arbitrario.

La esencia de todos los protocolos de ventana deslizante es que, en cualquier instante, el emisor mantiene un conjunto de números de secuencia que corresponde a las tramas que tiene permitido enviar. Se dice que estas tramas caen dentro de la **ventana emisora**. De manera similar, el receptor mantiene una **ventana receptora** correspondiente al conjunto de tramas que tiene permitido aceptar. La ventana del emisor y la del receptor no necesitan tener los mismos límites inferior y superior, ni siquiera el mismo tamaño. En algunos protocolos las ventanas son de tamaño fijo, pero en otros pueden aumentar o reducir su tamaño con el transcurso del tiempo, a medida que se envían y reciben las tramas.

Aunque estos protocolos dan a la capa de enlace de datos mayor libertad en cuanto al orden en que puede enviar y recibir tramas, hemos conservado decididamente el requerimiento de que el protocolo debe entregar los paquetes a la capa de red del destino en el mismo orden en que se pasaron a la capa de enlace de datos de la máquina emisora. Tampoco hemos cambiado el requerimiento de que el canal físico de comunicación sea de "tipo alambre"; es decir, que debe entregar todas las tramas en el orden en que fueron enviadas.

Los números de secuencia en la ventana del emisor representan las tramas que se han enviado, o que se pueden enviar pero aún no se ha confirmado su recepción. Cada vez que llega un paquete nuevo de la capa de red, se le asigna el siguiente número secuencial más alto y el extremo superior de la ventana avanza en uno. Cuando llega una confirmación de recepción, el extremo inferior avanza en uno. De esta manera, la ventana mantiene en forma continua una lista de tramas sin confirmación de recepción. En la figura 3-15 se muestra un ejemplo.

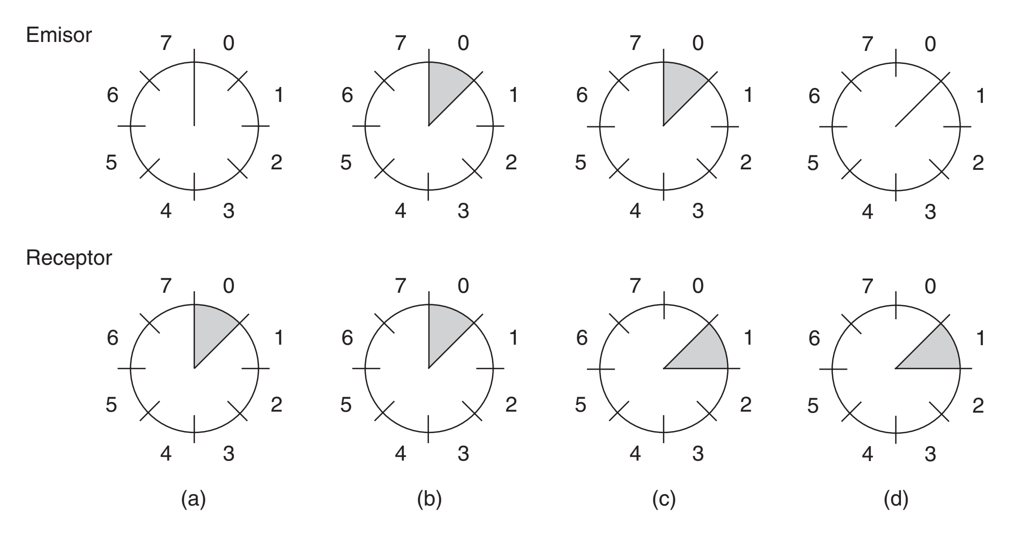


Figura 3-15. Una ventana deslizante de tamaño 1, con un número de secuencia de 3 bits. (a) Al inicio. (b) Después de enviar la primera trama. (c) Después de recibir la primera trama. (d) Después de recibir la primera confirmación de recepción.

Debido a que las tramas que están en la ventana del emisor se pueden perder o dañar en tránsito, el emisor debe mantener todas estas tramas en su memoria para su posible retransmisión. Por lo tanto, si el tamaño máximo de la ventana es *n*, el emisor necesita *n* búferes para contener las tramas sin confirmación de recepción. Si la ventana llega a crecer a su tamaño máximo, la capa de enlace de datos emisora deberá hacer que la capa de red se detenga hasta que se libere otro búfer.

La ventana de la capa de enlace de datos receptora corresponde a las tramas que puede aceptar. Toda trama que caiga dentro de la ventana se colocará en el búfer del receptor. Cuando se reciba una trama cuyo número de secuencia sea igual al extremo inferior de la ventana, se pasará a la capa de red y la ventana se desplazará una posición. Cualquier trama que caiga fuera de la ventana se desechará. En todos estos casos se genera una confirmación de recepción subsiguiente, de manera que el emisor pueda averiguar cómo proceder. Cabe mencionar que un tamaño de ventana de 1 significa que la capa de enlace de datos sólo acepta tramas en orden, pero con ventanas más grandes esto no es así. En contraste, la capa de red siempre recibe los datos en el orden correcto, sin importar el tamaño de la ventana de la capa de enlace de datos.

En la figura 3-15 se muestra un ejemplo con un tamaño máximo de ventana de 1. En un principio no hay tramas pendientes, por lo que los extremos inferior y superior de la ventana del emisor son iguales, pero a medida que pasa el tiempo, la situación progresa como se muestra. A diferencia de la ventana del emisor, la ventana del receptor siempre permanece en su tamaño inicial, y se desplaza a medida que se acepta la siguiente trama y se entrega a la capa de red.

3.4.1 Un protocolo de ventana deslizante de un bit

Antes de tratar el caso general, examinemos un protocolo de ventana deslizante con un tamaño máximo de ventana de 1. Tal protocolo utiliza parada y espera, ya que el emisor envía una trama y espera su confirmación de recepción antes de transmitir la siguiente.

En la figura 3-16 se describe dicho protocolo. Como los demás, comienza por definir algunas variables. *Next_frame_to_send* indica qué trama está tratando de enviar el emisor. Asimismo, *frame_expected* indica qué trama espera el receptor. En ambos casos, 0 y 1 son las únicas posibilidades.

En circunstancias normales, una de las dos capas de enlace de datos es la que comienza a transmitir la primera trama. En otras palabras, sólo uno de los programas de capa de enlace de datos debe contener las llamadas de procedimiento *to_physical_layer* y *start_timer* fuera del ciclo principal. La máquina que inicia obtiene el primer paquete de su capa de red, construye una trama a partir de él y la envía. Al llegar esta (o cualquier) trama, la capa de enlace de datos del receptor la verifica para saber si es un duplicado, igual que en el protocolo 3. Si la trama es la esperada, se pasa a la capa de red y la ventana del receptor se recorre hacia arriba.

El campo de confirmación de recepción contiene el número de la última trama recibida sin error. Si este número concuerda con el de secuencia de la trama que está tratando de enviar el emisor, éste sabe que ha terminado con la trama almacenada en el búfer (*buffer*) y que puede obtener el siguiente paquete de su capa de red. Si el número de secuencia no concuerda, debe seguir tratando de enviar la misma trama. Cada vez que se recibe una trama, también se regresa una.

Ahora examinemos el protocolo 4 para ver qué tan flexible es ante circunstancias patológicas. Suponga que la computadora *A* trata de enviar su trama 0 a la computadora *B* y que *B* trata de enviar su trama 0 a *A*. Suponga que *A* envía una trama a *B*, pero que el intervalo de temporización de *A* es un poco corto. En consecuencia, el temporizador de *A* se podría agotar repetidamente, enviando una serie de tramas idénticas, todas con $seq = 0$ y $ack = 1$.

Cuando llegue la primera trama válida a la computadora *B*, se aceptará y *frame_expected* se establecerá en 1. Todas las tramas subsiguientes serán rechazadas, puesto que ahora *B* espera tramas con el

```
/* El protocolo 4 (ventana deslizante) es bidireccional. */

#define MAX_SEQ 1                                    /* debe ser 1 para el protocolo 4 */
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"

void protocol4 (void)
{
seq_nr next_frame_to_send;                           /* sólo 0 o 1 */
seq_nr frame_expected;                               /* sólo 0 o 1 */
frame r, s;                                          /* variables de trabajo */
packet buffer;                                       /* paquete actual que se envía */
event_type event;

next_frame_to_send = 0;                              /* siguiente trama del flujo de salida */
frame_expected = 0;                                  /* próxima trama esperada */
from_network_layer(&buffer);                         /* obtiene un paquete de la capa de red */
s.info = buffer;                                     /* se prepara para enviar la trama inicial */
s.seq = next_frame_to_send;                          /* inserta el número de secuencia en la trama */
s.ack = 1 – frame_expected;                          /* confirmación de recepción superpuesta */
to_physical_layer(&s);                               /* transmite la trama */
start_timer(s.seq);                                  /* inicia el temporizador */

while (true){
    wait_for_event(&event);                          /* frame_arrival, cksum_err o timeout */
    if (event == frame_arrival){                     /* ha llegado una trama sin daño. */
      from_physical_layer(&r);                       /* la obtiene */
      if(r.seq == frame_expected) {                  /* maneja flujo de tramas de entrada. */
          to_network_layer(&r.info);                 /* pasa el paquete a la capa de red */
          inc(frame_expected);                       /* invierte el siguiente número de secuencia esperado */
      }
      if(r.ack == next_frame_to_send){               /* maneja flujo de tramas de salida. */
          stop_timer(r.ack);                         /* desactiva el temporizador */
          from_network_layer(&buffer);               /* obtiene un nuevo paquete de la capa de red */
          inc(next_frame_to_send);                   /* invierte el número de secuencia del emisor */
      }
    }
    s.info = buffer;                                 /* construye trama de salida */
    s.seq = next_frame_to_send;                      /* le inserta el número de secuencia */
    s.ack = 1 – frame_expected;                      /* número de secuencia de la última trama recibida */
    to_physical_layer(&s);                           /* transmite una trama */
    start_timer(s.seq);                              /* inicia el temporizador */
 }
}
```

Figura 3-16. Protocolo de ventana deslizante de 1 bit.

número de secuencia 1, no 0. Además, dado que los duplicados tienen $ack = 1$ y B aún está esperando una confirmación de recepción de 0, B no extraerá un nuevo paquete de su capa de red.

Cada vez que llegue un duplicado rechazado, B enviará a A una trama que contenga $seq = 0$ y $ack = 0$. Tarde o temprano una de éstas llegará correctamente a A y hará que empiece a enviar el siguiente paquete. Ninguna combinación de tramas perdidas o temporizadores que expiren antes de tiempo puede hacer que el protocolo entregue paquetes duplicados a cualquiera de las capas de red, ni que omita un paquete, ni que entre en un bloqueo irreversible. El protocolo es correcto.

Sin embargo, para demostrar qué tan sutiles pueden ser las interacciones entre los protocolos, cabe mencionar que si ambos lados envían de manera simultánea un paquete inicial, surge una situación pecu-

liar. En la figura 3-17 se muestra este problema de sincronización. En la parte (a) se muestra la operación normal del protocolo. En (b) se ilustra la peculiaridad. Si *B* espera la primera trama de *A* antes de enviar la suya, la secuencia es como se muestra en (a), y se aceptan todas las tramas.

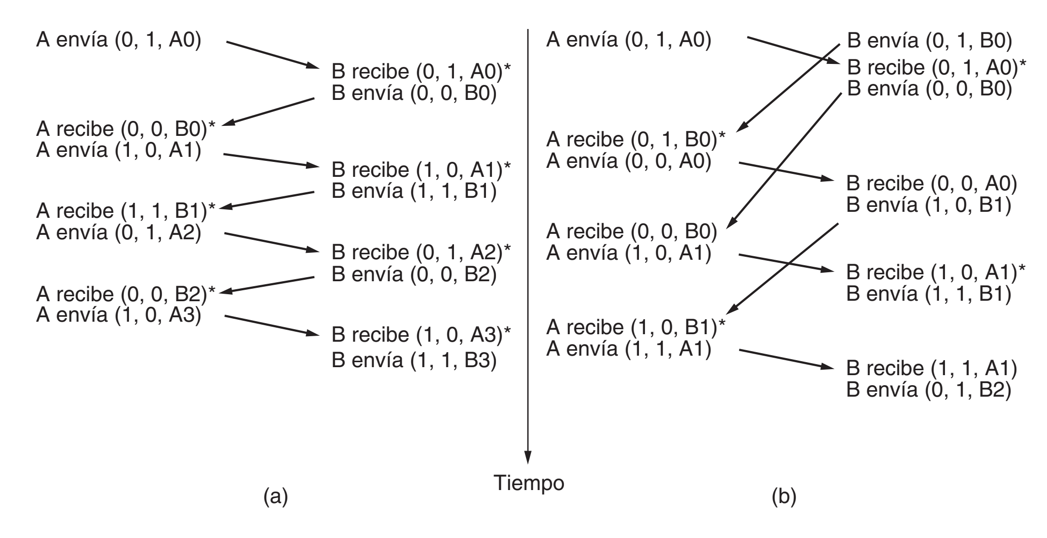


Figura 3-17. Dos escenarios para el protocolo 4. (a) Caso normal. (b) Caso anormal. La notación es (secuencia, confirmación de recepción, número de paquete). Un asterisco indica el lugar en que una capa de red acepta un paquete.

Pero si A y B inician la comunicación simultáneamente, se cruzan sus primeras tramas y las capas de enlace de datos entran en la situación (b). En (a) cada trama que llega trae un paquete nuevo para la capa de red; no hay duplicados. En (b) la mitad de las tramas contienen duplicados, aun cuando no hay errores de transmisión. Pueden ocurrir situaciones similares como resultado de la expiración prematura de temporizadores, incluso cuando sea evidente que un lado empezó primero. De hecho, si ocurren varias expiraciones prematuras de temporizadores, tal vez las tramas se envíen dos o tres veces, lo cual representa un desperdicio del valioso ancho de banda.

3.4.2 Un protocolo que utiliza retroceso N

Hasta ahora hemos supuesto que el tiempo de transmisión requerido para que una trama llegue al receptor más el necesario para que la confirmación de recepción regrese es insignificante. A veces esta suposición es totalmente falsa. En estas situaciones el tiempo de viaje de ida y vuelta prolongado puede tener implicaciones importantes para la eficiencia de la utilización del ancho de banda. Por ejemplo, considere un canal de satélite de 50 kbps con un retardo de propagación de ida y vuelta de 500 mseg. Imagine que intentamos utilizar el protocolo 4 para enviar tramas de 1000 bits por medio del satélite. En $t = 0$, el emisor empieza a enviar la primera trama. En $t = 20$ mseg la trama se ha enviado por completo. En el mejor de los casos (sin esperas en el receptor y con una trama de confirmación de recepción corta), no es sino hasta $t = 270$ mseg que la trama ha llegado por completo al receptor, y no es sino hasta $t = 520$ mseg que ha llegado la confirmación de recepción de regreso al emisor. Esto implica que el emisor estuvo bloqueado durante 500/520 o 96% del tiempo. En otras palabras, sólo se usó 4% del ancho de banda disponible. Sin duda, la combinación de un tiempo de tránsito grande, un alto ancho de banda y una longitud de tramas corta es desastrosa en términos de eficiencia.

El problema antes descrito puede considerarse como una consecuencia de la regla que requiere que el emisor espere una confirmación de recepción antes de enviar otra trama. Si relajamos esa restricción, podremos obtener una mejor eficiencia. Básicamente la solución está en permitir que el emisor envíe hasta w tramas antes de bloquearse, en lugar de que sólo envíe 1. Con una selección suficientemente grande de w, el emisor podrá transmitir tramas en forma continua, ya que las confirmaciones de recepción llegarán para las tramas anteriores antes de que la ventana se llene, lo cual evitará que el emisor se bloquee.

Para encontrar un valor apropiado para w necesitamos saber cuántas tramas pueden caber dentro del canal mientras se propagan del emisor al receptor. Esta capacidad se determina mediante el ancho de banda en bits/seg, multiplicado por el tiempo de tránsito en un sentido, mejor conocido como **producto de ancho de banda-retardo** del enlace. Podemos dividir esta cantidad entre el número de bits en una trama para expresar el producto como un número de tramas. Llamemos a esta cantidad BD. Entonces, w debe ser establecido a $2BD + 1$. El doble del producto ancho de banda-retardo es el número de tramas que pueden quedar pendientes si el emisor envía en forma continua tramas cuando se considera el tiempo de ida y vuelta para recibir una confirmación de recepción. El "$+1$" se debe a que no se enviará una trama de confirmación de recepción sino hasta recibir una trama completa.

Para el enlace de ejemplo con un ancho de banda de 50 kbps y un tiempo de tránsito en un sentido de 250 mseg, el producto de ancho de banda-retardo es de 12.5 kbit o 12.5 tramas de 1 000 bits cada una. Entonces, $2BD + 1$ es 26 tramas. Suponga que el emisor empieza a enviar la trama 0 como antes y que envía una nueva trama cada 20 mseg. Para cuando termine de enviar 26 tramas en $t = 520$ mseg, apenas sí llegará la confirmación de recepción de la trama 0. A partir de entonces, las confirmaciones de recepción llegarán cada 20 mseg, por lo que el emisor siempre tendrá permiso de continuar justo cuando lo necesite. De aquí en adelante siempre habrá 25 o 26 tramas pendientes de confirmación de recepción. Dicho de otra manera, el tamaño máximo de la ventana del emisor es de 26.

Para tamaños de ventana más pequeños, el uso del enlace será de menos de 100% debido a que el emisor estará bloqueado algunas veces. Podemos escribir la utilización como la fracción de tiempo que el emisor no está bloqueado:

$$\text{utilización del enlace} \leq \frac{w}{1 + 2BD}$$

Este valor es un límite superior ya que no considera ningún tiempo de procesamiento de tramas y supone que la trama de confirmación de recepción tiene una longitud de cero, puesto que generalmente es corta. La ecuación muestra la necesidad de tener una ventana w grande siempre que el producto ancho de banda-retardo también lo sea. Si el retardo es alto, el emisor agotará rápidamente su ventana incluso para un ancho de banda moderado, como en el ejemplo del satélite. Si el ancho de banda es alto, incluso para un retardo moderado, el emisor agotará su ventana con rapidez, a menos que tenga una ventana grande (por ejemplo, un enlace de 1 Gbps con un retraso de 1 mseg contiene 1 megabit). Con el protocolo de parada y espera en el cual $w = 1$, si hay incluso una trama equivalente al retardo de propagación, la eficiencia será menor a 50%.

Esta técnica de mantener varias tramas en movimiento es un ejemplo de **canalización (***pipeling***)**. La canalización de tramas a través de un canal de comunicación no confiable presenta problemas serios. Primero, ¿qué ocurre si una trama a la mitad de un flujo extenso se daña o pierde? Llegarán grandes cantidades de tramas sucesivas al receptor antes de que el emisor se entere de que algo anda mal. Cuando llega una trama dañada al receptor es obvio que debe descartarse pero, ¿qué debe hacer el receptor con todas las tramas correctas que le siguen? Recuerde que la capa de enlace de datos del receptor está obligada a entregar paquetes a la capa de red en secuencia.

Hay dos métodos básicos disponibles para tratar con los errores en presencia de la canalización, ambos se muestran en la figura 3-18.

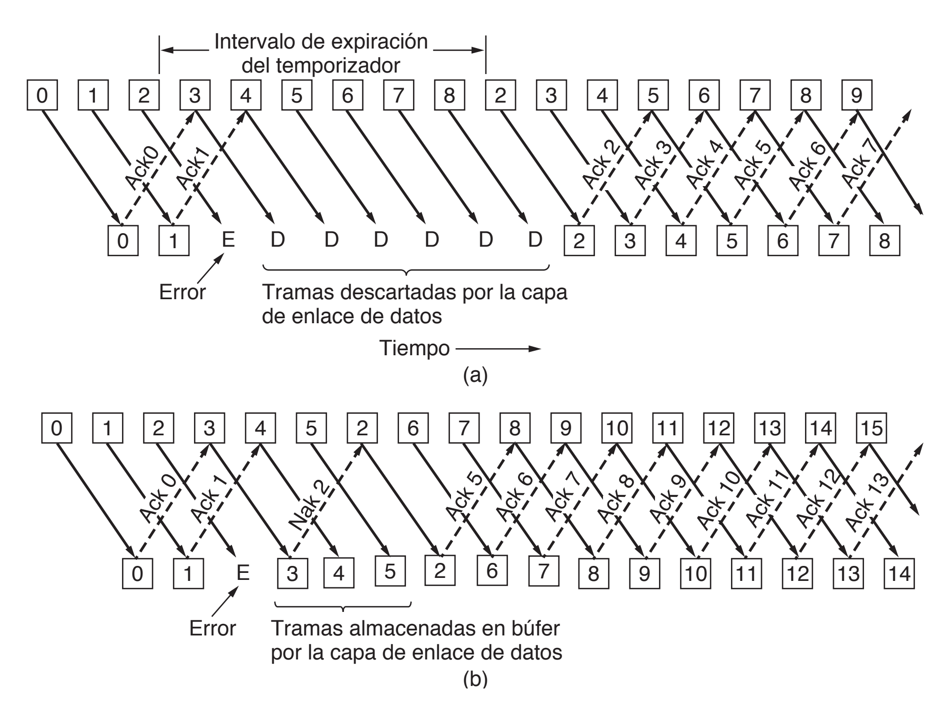


Figura 3-18. Canalización y recuperación de errores. Efecto de un error cuando (a) el tamaño de la ventana del receptor es 1 y (b) el tamaño de la ventana del receptor es grande.

Una de las opciones, llamada **retroceso n**, es que el receptor simplemente descarte todas las tramas subsecuentes, sin enviar confirmaciones de recepción para las tramas descartadas. Esta estrategia corresponde a una ventana de recepción de tamaño 1. En otras palabras, la capa de enlace de datos se niega a aceptar cualquier trama excepto la siguiente que debe entregar a la capa de red. Si la ventana del emisor se llena antes de que expire el temporizador, el canal comenzará a vaciarse. En algún momento, el emisor terminará de esperar y retransmitirá en orden todas las tramas cuya recepción aún no se haya confirmado, comenzando por la trama dañada o perdida. Esta estrategia puede desperdiciar bastante ancho de banda si la tasa de error es alta.

En la figura 3-18(b) podemos ver el retroceso n para el caso en que la ventana del receptor es grande. Las tramas 0 y 1 se reciben y confirman de manera correcta. Sin embargo, la trama 2 se daña o pierde. El emisor, sin saber sobre este problema, continúa enviando tramas hasta que expira el temporizador para la trama 2. Después retrocede a la trama 2 y comienza con ella, enviando nuevamente las tramas 2, 3, 4, etcétera.

La otra estrategia general para el manejo de errores cuando las tramas se colocan en canalizaciones se conoce como **repetición selectiva**. Cuando se utiliza, si se recibe una trama dañada se descarta, pero las tramas en buen estado que se reciban después de ésa se aceptan y almacenan en el búfer. Cuando expira el temporizador del emisor, sólo se retransmite la última trama sin confirmación de recepción. Si la trama llega correctamente, el receptor puede entregar a la capa de red, en secuencia, todas las tramas que ha almacenado en el búfer. La repetición selectiva corresponde a una ventana del receptor mayor a 1. Este método puede requerir grandes cantidades de memoria en la capa de enlace de datos, si la ventana es grande.

A menudo, la repetición selectiva se combina con el hecho de que el receptor envíe una confirmación de recepción negativa NAK (del inglés *negative aknowledgement*) al detectar un error; por ejemplo, cuando recibe un error de suma de verificación o una trama fuera de secuencia. Las NAK estimulan la retransmisión antes de que el temporizador correspondiente expire y, por lo tanto, mejoran el rendimiento.

En la figura 3-18(b), las tramas 0 y 1 se vuelven a recibir y confirmar correctamente, pero la trama 2 se pierde. Cuando la trama 3 llega al receptor, su capa de enlace de datos observa que falta una trama, por lo que regresa una NAK para la trama 2 pero almacena la trama 3 en el búfer. Cuando llegan las tramas 4 y 5, la capa de enlace de datos también las almacena en el búfer, en lugar de pasarlas a la capa de red. En algún momento la NAK 2 llega al emisor, que inmediatamente reenvía la trama 2. Cuando ésta llega, la capa de enlace de datos tiene las tramas 2, 3, 4 y 5, y puede pasarlas todas a la capa de red en el orden correcto. También puede confirmar la recepción de todas las tramas incluyendo la 5, como se muestra en la figura. Si la NAK se perdiera, en algún momento el temporizador del emisor expiraría para la trama 2 y la enviaría (sólo a ella) por su cuenta, pero eso podría tardar un poco más.

Estos dos métodos alternativos son concesiones entre el uso eficiente del ancho de banda y el espacio de búfer en la capa de enlace de datos. Dependiendo de qué recurso sea más valioso, se puede utilizar uno u otro. En la figura 3-19 se muestra un protocolo de retroceso n en el que la capa de enlace de datos del receptor sólo acepta tramas en orden; las tramas que siguen después de un error se descartan. En este protocolo hemos omitido por primera vez el supuesto de que la capa de red siempre tiene que enviar un suministro infinito de paquetes. Cuando la capa de red tiene un paquete que desea enviar, puede hacer que ocurra un evento *network_layer_ready*. Sin embargo, para mantener el límite de control de flujo en la ventana del emisor o el número de tramas sin confirmación de recepción pendientes en cualquier momento, la capa de enlace de datos debe ser capaz de prohibir a la capa de red que la moleste con más trabajo. Los procedimientos de biblioteca *enable_network_layer* y *disable_network_layer* llevan a cabo esta tarea.

El número máximo de tramas que pueden estar pendientes en cualquier instante no es igual que el tamaño del espacio del número de secuencia. Para el retroceso n puede haber *MAX_SEQ* tramas pendientes en cualquier instante, aun cuando haya *MAX_SEQ* + 1 números de secuencia diferentes (que son 0, 1,..., *MAX_SEQ*). En el siguiente protocolo, repetición selectiva, veremos una restricción aún mayor. Para ver por qué es necesaria esta restricción, considere el siguiente escenario en donde *MAX_SEQ* = 7.

1. El emisor envía las tramas 0 a 7.
2. Llega al emisor una confirmación de recepción superpuesta para la trama 7.
3. El emisor envía otras ocho tramas, de nuevo con los números de secuencia 0 a 7.
4. Ahora llega otra confirmación de recepción superpuesta para la trama 7.

La pregunta es: ¿llegaron con éxito las ocho tramas que correspondían al segundo bloque o se perdieron (contando como pérdidas las tramas descartadas después de un error)? En ambos casos el receptor podría estar enviando la trama 7 como la confirmación de recepción. El emisor no tiene manera de saberlo. Por esta razón, el número máximo de tramas pendientes se debe restringir a *MAX_SEQ*.

Aunque el protocolo 5 no pone en el búfer las tramas que llegan después de un error, no escapa del problema de los búferes por completo. Dado que un emisor tal vez tenga que retransmitir en un futuro todas las tramas no confirmadas, debe retener todas las tramas transmitidas hasta saber con certeza que el receptor las aceptó. Cuando llega una confirmación de recepción para la trama n, las tramas $n - 1$, $n - 2$ y demás se confirman también de manera automática. Este tipo de confirmación de recepción se llama **confirmación de recepción acumulativa**. Esta propiedad es importante cuando algunas de las tramas previas portadoras de confirmaciones de recepción se perdieron o dañaron. Cuando llega una confirmación de recepción, la capa de enlace de datos verifica si se pueden liberar búferes. Si esto es posible

```
/* El protocolo 5 (retroceso n) permite múltiples tramas pendientes. El emisor podría enviar hasta MAX_SEQ
   tramas sin esperar una confirmación de recepción. Además, a diferencia de los protocolos anteriores, no
   existe el supuesto de que la capa de red debe tener siempre un paquete nuevo. En vez de ello, la capa de red
   activa un evento network_layer_ready cuando hay un paquete por enviar. */

#define MAX_SEQ 7
typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready} event_type;
#include "protocol.h"

static boolean between(seq_nr a, seq_nr b, seq_nr c)
{
/* Devuelve true si a <=b < c de manera circular, y false en caso contrario. */
if (((a <= b) && (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a)))
      return(true);
  else
      return(false);
}

static void send_data(seq_nr frame_nr, seq_nr frame_expected, packet buffer[])
{
/* Elabora y envía una trama de datos. */
frame s;                                    /* variable de trabajo */

  s.info = buffer[frame_nr];                /* inserta el paquete en la trama */
  s.seq = frame_nr;                         /* inserta un número de secuencia en la trama */
  s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1); /* ack superpuesta*/
  to_physical_layer(&s);                    /* transmite la trama */
  start_timer(frame_nr);                    /* inicia la ejecución del temporizador */
}

void protocol5(void)
{
  seq_nr next_frame_to_send;                /* MAX_SEQ > 1; se usa para flujo de salida */
  seq_nr ack_expected;                      /* la trama más antigua no confirmada hasta el momento*/
  seq_nr frame_expected;                    /* siguiente trama esperada en el flujo de entrada */
  frame r;                                  /* variable de trabajo */
  packet buffer[MAX_SEQ + 1];               /* búferes para el flujo de salida */
  seq_nr nbuffered;                         /* número de búferes de salida actualmente en uso */
  seq_nr i;                                 /* se usa para indexar en el arreglo de búferes */
  event_type event;

  enable_network_layer();                   /* permite eventos network_layer_ready */
  ack_expected = 0;                         /* siguiente ack esperada en el flujo de entrada */
  next_frame_to_send = 0;                   /* siguiente trama de salida */
  frame_expected = 0;                       /* número de trama de entrada esperada */
  nbuffered = 0;                            /* al inicio no hay paquetes en el búfer */

while (true) {
   wait_for_event(&event);                  /* cuatro posibilidades: vea event_type al principio */
switch(event) {
   case network_layer_ready:                /* la capa de red tiene un paquete para enviar */
      /* Acepta, guarda y transmite una trama nueva. */
      from_network_layer(&buffer[next_frame_to_send]); /* obtiene un paquete nuevo */
      nbuffered = nbuffered + 1;            /* expande la ventana del emisor */
      send_data(next_frame_to_send, frame_expected, buffer); /* transmite la trama */
      inc(next_frame_to_send);              /* avanza el límite superior de la ventana del emisor */
      break;
```

(Continúa)

(Continuación)

```
  case frame_arrival:                     /* ha llegado una trama de datos o de control */
    from_physical_layer(&r);              /* obtiene una trama entrante de la capa física */

    if (r.seq == frame_expected) {
      /* Las tramas se aceptan sólo en orden. */
      to_network_layer(&r.info);          /* pasa el paquete a la capa de red */
      inc(frame_expected);                /* avanza el límite inferior de la ventana del receptor */
    }

    /* Ack n implica n - 1, n - 2, etc. Verificar esto. */
    while (between(ack_expected, r.ack, next_frame_to_send)) {
      /* Maneja la ack superpuesta. */
      nbuffered = nbuffered - 1;          /* una trama menos en el búfer */
      stop_timer(ack_expected);           /* la trama llegó intacta; detener el temporizador */
      inc(ack_expected);                  /* contrae la ventana del emisor */
    }
    break;

  case cksum_err: break;                  /* ignora las tramas erróneas */

  case timeout:                           /* problemas; retransmite todas las tramas pendientes */
    next_frame_to_send = ack_expected; /* aquí inicia la retransmisión */
    for (i = 1; i <= nbuffered; i++) {
      send_data(next_frame_to_send, frame_expected, buffer); /* reenvía trama  */
      inc(next_frame_to_send);            /* se prepara para enviar la siguiente */
    }
  }
  if (nbuffered < MAX_SEQ)
      enable_network_layer();
  else
      disable_network_layer();
  }
}
```

Figura 3-19. Un protocolo de ventana deslizante con retroceso n.

(es decir, hay espacio disponible en la ventana), a una capa de red bloqueada se le puede permitir que produzca más eventos *network_layer_ready*.

Para este protocolo damos por hecho que siempre hay tráfico de regreso en el que se pueden superponer confirmaciones de recepción. El protocolo 4 no necesita este supuesto debido a que envía una trama cada vez que recibe una, incluso si ya la ha enviado. En el siguiente protocolo resolveremos el problema del tráfico de un solo sentido de una forma elegante.

Como el protocolo 5 tiene múltiples tramas pendientes, por lógica necesita múltiples temporizadores, uno por cada trama pendiente. El temporizador de cada trama expira de manera independiente a los de las otras tramas. Sin embargo, todos estos temporizadores se pueden simular fácilmente en software, mediante el uso de un solo reloj de hardware que produzca interrupciones en forma periódica. Los temporizadores pendientes de expirar forman una lista enlazada, en la que cada nodo contiene la cantidad de pulsos de reloj que restan para que expire el temporizador, la trama temporizada y un apuntador al siguiente nodo.

Para ver cómo se pueden implementar los temporizadores, considere el ejemplo de la figura 3-20(a). Suponga que el reloj pulsa una vez cada 1 mseg. Al principio, la hora real es 10:00:00.000; hay tres temporizadores por expirar pendientes, a las 10:00:00.005, 10:00:00.013 y 10:00:00.019. Cada vez que pulsa el reloj de hardware, se actualiza el tiempo real y se decrementa el contador de pulsos a la cabeza de la

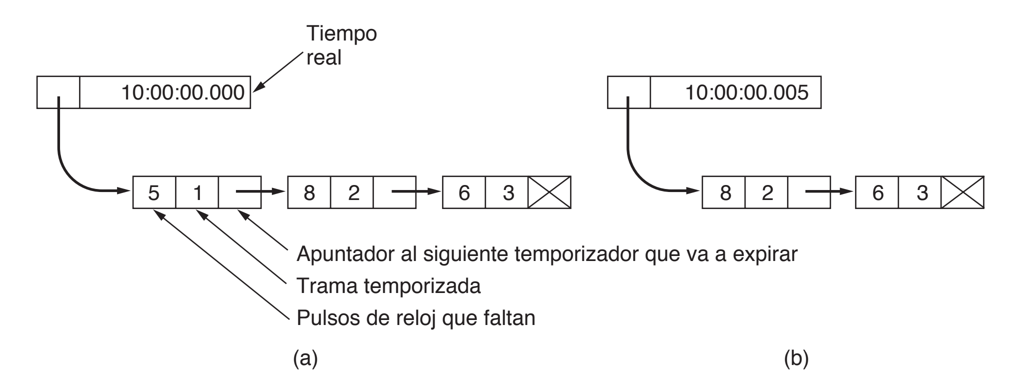


Figura 3-20. Simulación de varios temporizadores en software. (a) Los temporizadores que van a expirar puestos en cola. (b) La situación después de que expira el primer temporizador.

lista. Cuando el contador de pulsos llega a cero, un temporizador expira y se retira el nodo de la lista, como se muestra en la figura 3-20(b). Aunque esta organización requiere que se analice la lista al llamar a *start_timer* o a *stop_timer*, no requiere mucho trabajo por pulso de reloj. En el protocolo 5 estas dos rutinas tienen un parámetro que indica la trama a temporizar.

3.4.3 Un protocolo que usa repetición selectiva

El protocolo de retroceso n funciona bien si los errores son poco frecuentes, pero si la línea es mala se desperdicia mucho ancho de banda en las tramas que se retransmiten. El protocolo de repetición selectiva es una estrategia alterna para que el receptor acepte y coloque en búferes las tramas que llegan después de una trama dañada o perdida.

En este protocolo, tanto el emisor como el receptor mantienen una ventana de números de secuencia pendientes y aceptables, respectivamente. El tamaño de la ventana del emisor comienza en 0 y crece hasta un máximo predefinido. Por el contrario, la ventana del receptor siempre es de tamaño fijo e igual al máximo predeterminado. El receptor tiene un búfer reservado para cada número de secuencia dentro de su ventana fija. Cada búfer tiene un bit asociado (*arrived*), el cual indica si el búfer está lleno o vacío. Cada vez que llega una trama, la función *between* verifica su número de secuencia para ver si cae dentro de la ventana. De ser así, y si todavía no se recibe, se acepta y almacena. Esta acción se lleva a cabo sin importar si la trama contiene o no el siguiente paquete que espera la capa de red. Claro que se debe mantener dentro de la capa de enlace de datos sin pasarla a la capa de red hasta que se hayan entregado todas las tramas de menor número a la capa de red en el orden correcto. En la figura 3-21 se presenta un protocolo que usa este algoritmo.

La recepción no secuencial introduce limitaciones adicionales en los números de secuencia de tramas, que no se presentan en los protocolos en los que las tramas sólo se aceptan en orden. Podemos ilustrar el problema fácilmente con un ejemplo. Suponga que tenemos un número de secuencia de 3 bits, de modo que se permita al emisor transmitir hasta siete tramas antes detener que esperar una confirmación de recepción. En un principio las ventanas del emisor y del receptor están como se muestra en la figura 3-22(a). Ahora el emisor transmite las tramas 0 a 6. La ventana del receptor le permite aceptar cualquier trama con un número de secuencia entre 0 y 6, inclusive. Las siete tramas llegan correctamente, por lo que el receptor confirma su recepción y avanza su ventana para permitir la recepción de la trama 7, 0, 1, 2, 3, 4 o 5, como se muestra en la figura 3-22(b). Los siete búfer se marcan como vacíos.

```
/* El protocolo 6 (repetición selectiva) acepta tramas en desorden y pasa paquetes en orden a la capa de red.
   Cada trama pendiente tiene un temporizador asociado. Cuando el temporizador expira, a diferencia de lo que
   ocurre en el protocolo 5, sólo se retransmite esa trama y no todas las que están pendientes. */

#define MAX_SEQ 7                                  /* debe ser 2^n – 1 */
#define NR_BUFS ((MAX_SEQ + 1)/2)
typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready, ack_timeout} event_type;
#include "protocol.h"
boolean no_nak = true;                             /* aún no se ha enviado un nak */
seq_nr oldest_frame = MAX_SEQ + 1;                 /* el valor inicial es sólo para el simulador */

static boolean between(seq_nr a, seq_nr b, seq_nr c)
{
/* Parecido a lo que ocurre en el protocolo 5, pero más corto y confuso. */
return ((a <= b) && (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a));
}

static void send_frame(frame_kind fk, seq_nr frame_nr, seq_nr frame_expected, packet buffer[])
{
/* Construye y envía una trama de datos, ack o nak. */
frame s;                                           /* variable de trabajo */

 s.kind = fk;                                      /* kind == datos, ack o nak */
 if (fk == data) s.info = buffer[frame_nr % NR_BUFS];
 s.seq = frame_nr;                                 /* sólo tiene importancia para las tramas de datos */
 s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1);
 if (fk == nak) no_nak = false;                    /* un nak por trama, por favor */
 to_physical_layer(&s);                            /* transmite la trama */
 if (fk == data) start_timer(frame_nr % NR_BUFS);
 stop_ack_timer();                                 /* no se necesita para tramas ack independientes */
}

void protocol6(void)
{
 seq_nr ack_expected;                              /* límite inferior de la ventana del emisor */
 seq_nr next_frame_to_send;                        /* límite superior de la ventana del emisor + 1 */
 seq_nr frame_expected;                            /* límite inferior de la ventana del receptor */
 seq_nr too_far;                                   /* límite superior de la ventana del receptor + 1 */
 int i;                                            /* índice en el grupo de búferes */
 frame r;                                          /* variable de trabajo*/
 packet out_buf[NR_BUFS];                          /* búferes para el flujo de salida */
 packet in_buf[NR_BUFS];                           /* búferes para el flujo de entrada */
 boolean arrived[NR_BUFS];                         /* mapa de bits de entrada */
 seq_nr nbuffered;                                 /* cuántos búferes de salida se utilizan actualmente */
 event_type event;

 enable_network_layer();                           /* inicializar */
 ack_expected = 0;                                 /* siguiente ack esperada en el flujo de entrada */
 next_frame_to_send = 0;                           /* número de la siguiente trama de salida */
 frame_expected = 0;
 too_far = NR_BUFS;
 nbuffered = 0;                                    /* al principio no hay paquetes en el búfer */
 for (i = 0; i < NR_BUFS; i++) arrived[i] = false;
 while (true) {
  wait_for_event(&event);                          /* cinco posibilidades: vea event_type al principio */
  switch(event) {
   case network_layer_ready:                       /* acepta, guarda y transmite una trama nueva */
```

(Continúa)

(Continuación)

```
        nbuffered = nbuffered + 1;                 /* expande la ventana */
        from_network_layer(&out_buf[next_frame_to_send % NR_BUFS]); /* obtiene un paquete nuevo */
        send_frame(data, next_frame_to_send, frame_expected, out_buf); /* transmite la trama */
        inc(next_frame_to_send);                   /* avanza el límite superior de la ventana */
        break;

     case frame_arrival:                           /* ha llegado una trama de datos o de control */
        from_physical_layer(&r);                   /* obtiene una trama entrante de la capa física */
        if (r.kind == data) {
          /* Ha llegado una trama no dañada. */
          if ((r.seq != frame_expected) && no_nak)
            send_frame(nak, 0, frame_expected, out_buf); else start_ack_timer();
          if (between(frame_expected, r.seq, too_far) && (arrived[r.seq%NR_BUFS]== false)) {
            /* Las tramas se podrían aceptar en cualquier orden. */
            arrived[r.seq % NR_BUFS] = true;       /* marca el búfer como lleno */
            in_buf[r.seq % NR_BUFS] = r.info;      /* inserta datos en el búfer */
            while (arrived[frame_expected % NR_BUFS]) {
              /* Pasa tramas y avanza la ventana. */
              to_network_layer(&in_buf[frame_expected % NR_BUFS]);
              no_nak = true;
              arrived[frame_expected % NR_BUFS] = false;
              inc(frame_expected);                 /* avanza el límite inferior de la ventana del receptor */
              inc(too_far);                        /* avanza el límite superior de la ventana del receptor */
              start_ack_timer();                   /* para saber si es necesaria una ack independiente */
            }
          }
        }
        if((r.kind==nak) && between(ack_expected,(r.ack+1)%(MAX_SEQ+1), next_frame_to_send))
          send_frame(data, (r.ack+1) % (MAX_SEQ + 1), frame_expected, out_buf);

        while (between(ack_expected, r.ack, next_frame_to_send)) {
          nbuffered = nbuffered - 1;               /* maneja la ack superpuesta */
          stop_timer(ack_expected % NR_BUFS);      /* la trama llega intacta */
          inc(ack_expected);                       /* avanza el límite inferior de la ventana del emisor */
        }
        break;
     case cksum_err:
        if (no_nak) send_frame(nak, 0, frame_expected, out_buf);  /* trama dañada */
        break;

     case timeout:
        send_frame(data, oldest_frame, frame_expected, out_buf); /* expiró el temporizador */
        break;

     case ack_timeout:
        send_frame(ack,0,frame_expected, out_buf); /* expiró el temporizador de ack; envía ack */
   }
   if (nbuffered < NR_BUFS) enable_network_layer(); else disable_network_layer();
 }
}
```

Figura 3-21. Protocolo de ventana deslizante con repetición selectiva.

En este momento, ocurre un desastre: un rayo cae en el poste telefónico y borra todas las confirmaciones de recepción. El protocolo debe ser capaz de operar en forma correcta a pesar de esto. En algún momento expira el temporizador del emisor y éste retransmite la trama 0. Cuando llega esta trama al receptor,

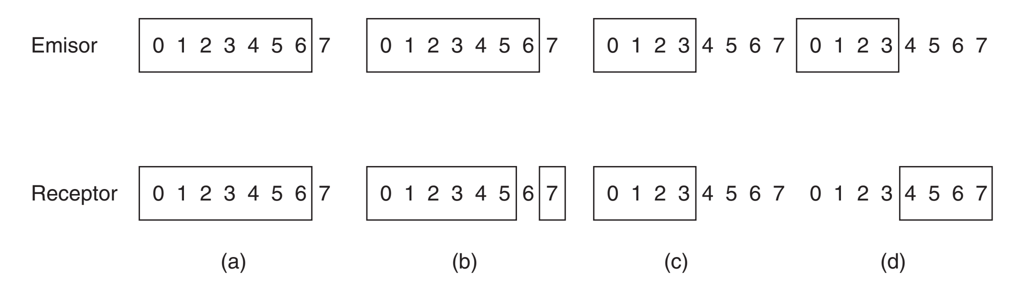


Figura 3-22. (a) Situación inicial con una ventana de tamaño 7. (b) Después de enviar y recibir 7 tramas sin regresar confirmaciones de recepción. (c) Situación inicial con un tamaño de ventana de 4. (d) Después de enviar y recibir 4 tramas sin regresar confirmaciones de recepción.

se efectúa una verificación para saber si está dentro de la ventana del mismo. Por desgracia, en la figura 3-22(b) la trama 0 está dentro de la nueva ventana, por lo que se acepta como una nueva trama. El receptor también envía una confirmación de recepción (superpuesta) para la trama 6, ya que se han recibido de la 0 a la 6.

El emisor está feliz de saber que todas las tramas que transmitió llegaron en forma correcta, por lo que avanza su ventana y de inmediato envía las tramas 7, 0, 1, 2, 3, 4 y 5. El receptor aceptará la trama 7 y su paquete se pasará directamente a la capa de red. Enseguida la capa de enlace de datos receptora verifica si ya tiene una trama 0 válida, descubre que sí y pasa el paquete anterior del búfer a la capa de red como si fuera el nuevo paquete. En consecuencia, la capa de red obtiene un paquete incorrecto y el protocolo fracasa.

La esencia del problema es que, una vez que el receptor ha avanzado su ventana, el nuevo intervalo de números de secuencia válidos se traslapa con el anterior. En consecuencia, el siguiente grupo de tramas podría contener tramas duplicadas (si se perdieron todas las confirmaciones de recepción) o nuevas (si se recibieron todas las confirmaciones de recepción). El pobre receptor no tiene manera de distinguir entre estos dos casos.

La salida de este dilema es asegurarse que, una vez que el emisor haya avanzado su ventana, no haya traslape con la ventana original. Para asegurarse de que no haya traslape, el tamaño máximo de la ventana debe ser cuando menos de la mitad del intervalo de los números de secuencia. Esta situación se muestra en las figuras 3-22(c) y 3-22(d). Con 3 bits, los números de secuencia varían de 0 a 7. Sólo debe haber cuatro tramas sin confirmación de recepción pendientes en cualquier instante. De esa forma, si el receptor acaba de aceptar las tramas 0 a 3 y ha avanzado su ventana para permitir la aceptación de las tramas 4 a 7, puede distinguir sin ambigüedades si las tramas subsiguientes son retransmisiones (0 a 3) o si son nuevas (4 a 7). En general, el tamaño de la ventana para el protocolo 6 será ($MAX_SEQ + 1$)/2.

Una pregunta interesante es: ¿cuántos búfer debe tener el receptor? En ninguna circunstancia podrá aceptar tramas cuyos números de secuencia estén por debajo del extremo inferior de la ventana o de las tramas cuyos números de secuencia estén por encima del extremo superior de la ventana. En consecuencia, el número de búfer necesarios es igual al tamaño de la ventana, no al intervalo de números de secuencia. En el ejemplo anterior de un número de secuencia de 3 bits, se requieren cuatro búfer numerados del 0 al 3. Al llegar la trama i, se coloca en el búfer i mod 4. Tenga en cuenta que, aun cuando i e ($i + 4$) mod 4 están "compitiendo" por el mismo búfer, nunca están dentro de la ventana al mismo tiempo, pues ello implicaría un tamaño de ventana de cuando menos 5.

Por la misma razón, el número de temporizadores requeridos es igual al número de búfer no al tamaño del espacio de secuencia. En efecto, hay un temporizador asociado a cada búfer. Cuando expira el temporizador, el contenido del búfer se retransmite.

El protocolo 6 también suaviza la suposición implícita de que el canal está fuertemente cargado. Hicimos esta suposición en el protocolo 5, cuando requeríamos que se enviaran tramas en dirección inversa para superponer las confirmaciones de recepción. Si el tráfico de regreso es ligero, las confirmaciones de recepción se pueden retener por un largo periodo, lo cual puede provocar problemas. En un caso extremo, si hay mucho tráfico en una dirección y no hay tráfico en la otra dirección, el protocolo se bloqueará cuando la ventana del emisor llegue a su máximo

Para suavizar esta suposición, se inicia un temporizador auxiliar mediante *start_ack_timer* después de que llega una trama de datos en la secuencia correcta. Si no se ha presentado tráfico de regreso antes de que expire este temporizador, se envía una trama de confirmación de recepción independiente. Una interrupción debida al temporizador auxiliar se denomina evento *ack_timeout*. Con este arreglo, ahora es posible el flujo de tráfico unidireccional, pues el que no haya tramas de datos de regreso a las que se puedan superponer las confirmaciones de recepción ya no es un obstáculo. Sólo existe un temporizador auxiliar; si se invoca a *start_ack_timer* mientras el temporizador está en funcionamiento, no tiene efecto. El temporizador no se restablece ni se extiende, ya que su propósito es proveer cierta tasa mínima de confirmaciones de recepción.

Es indispensable que el tiempo de expiración asociado al temporizador auxiliar sea notablemente más corto que el del temporizador usado para terminar las tramas de datos. Esta condición es necesaria para asegurarse de que la confirmación de recepción de una trama recibida correctamente llegue antes de que expire el temporizador de retransmisión de la trama y la retransmita.

El protocolo 6 utiliza una estrategia más eficiente que el protocolo 5 para manejar los errores. Cuando el receptor tiene razones para sospechar que ocurrió un error, envía al emisor una trama de confirmación de recepción negativa (NAK). Dicha trama es una solicitud de retransmisión de la trama especificada en la NAK. Hay dos casos en los que el receptor debe sospechar: cuando llega una trama dañada, o cuando llega una trama diferente de la esperada (pérdida potencial de la trama). Para evitar hacer múltiples solicitudes de retransmisión de la misma trama perdida, el receptor debe estar al tanto de si ya se ha enviado una NAK para una trama específica. La variable *no_nak* del protocolo 6 es *true* si no se ha enviado todavía ninguna NAK para *frame_expected*. Si la NAK se altera o se pierde no hay un daño real, pues de todas formas expirará el temporizador del emisor en un momento dado y retransmitirá la trama que se perdió. Si llega la trama equivocada después de haber enviado una NAK y que ésta se haya perdido, *no_nak* será *true* y se iniciará el temporizador auxiliar. Cuando expire, se enviará una ACK para resincronizar el estado actual del emisor con el del receptor.

En algunas situaciones, el tiempo requerido para que una trama se propague a su destino, se procese ahí y se devuelva la confirmación de recepción es (casi) constante. En estos casos, el emisor puede ajustar su temporizador para que apenas sea mayor que el intervalo esperado entre enviar una trama y la recepción de su confirmación. En este caso no se utilizan las NAK.

No obstante, en otros casos el tiempo puede ser muy variable. Por ejemplo, si el tráfico de regreso es esporádico, el tiempo antes de la confirmación de recepción será más corto cuando haya tráfico de regreso y más largo cuando no lo haya. El emisor se enfrenta a la elección entre establecer el intervalo a un valor pequeño (y arriesgarse a que haya retransmisiones innecesarias) o establecerlo a un valor grande (y estar inactivo por un largo periodo después de un error). Ambas opciones desperdician ancho de banda. En general, si la desviación estándar del intervalo de confirmación de recepción es grande en comparación con el intervalo mismo, el temporizador se establecerá a un valor "holgado" que sea conservador. Así, las NAK pueden acelerar de manera apreciable la retransmisión de tramas perdidas o dañadas.

Un aspecto muy relacionado con el asunto de la expiración de los temporizadores y las NAK es cómo determinar qué trama causó que expirara un temporizador. En el protocolo 5 siempre es *ack_expected*, puesto que es la más antigua. En el protocolo 6 no hay ninguna manera sencilla de determinar quién hizo que expirara el temporizador. Suponga que ya se transmitieron las tramas 0 a 4, de modo que la lista de tramas pendientes es 01234, en orden de la más antigua a la más nueva. Ahora imagine que expira el temporizador

de la trama 0, se transmite 5 (una trama nueva), expira el temporizador de 1, expira el temporizador de 2 y se transmite 6 (otra trama nueva). En este punto la lista de tramas pendientes es 3405126, de la más antigua a la más nueva. Si todo el tráfico de entrada (es decir, las tramas que llevan las confirmaciones de recepción) se pierde durante un momento, expirará el temporizador de las siete tramas pendientes en ese orden.

Para evitar que el ejemplo se complique aún más, no hemos mostrado la administración de los temporizadores. En cambio, sólo suponemos que la variable *oldest_ frame* se establece al momento en que expira el temporizador, para indicar qué trama hizo que expirara.

3.5 EJEMPLOS DE PROTOCOLOS DE ENLACE DE DATOS

En el interior de un solo edificio las redes LAN se utilizan mucho para interconectar computadoras, pero la mayoría de la infraestructura de redes de área amplia se basa en las líneas de punto a punto. En el capítulo 4 analizaremos las redes LAN. En esta sección examinaremos los protocolos de enlace de datos que se encuentran en las líneas de punto a punto en Internet, en dos situaciones comunes. La primera situación es cuando los paquetes se envían a través de enlaces de fibra óptica SONET en redes de área amplia. Estos enlaces se utilizan mucho, por ejemplo, para conectar enrutadores en las distintas ubicaciones de la red de un ISP.

La segunda situación para los enlaces de ADSL que operan en el lazo local de la red telefónica, en un extremo de Internet. Estos enlaces conectan a millones de individuos y negocios a Internet.

Internet necesita enlaces de punto a punto para estos usos, así como módems de marcación telefónica, líneas rentadas y módems de cable, etc. Un protocolo estándar llamado **PPP** (**Protocolo Punto a Punto**) se utiliza para enviar paquetes a través de estos enlaces. PPP se define en el RFC 1661 y se ha desarrollado más en RFC 1662 y otros RFC (Simpson, 1994a, 1994b). Los enlaces SONET y ADSL aplican PPP, sólo que en distintas formas.

3.5.1 Paquetes sobre SONET

SONET, que vimos en la sección 2.6.4, es el protocolo de capa física que se utiliza con más frecuencia sobre los enlaces de fibra óptica de área amplia que constituyen la espina dorsal de las redes de comunicaciones, incluyendo el sistema telefónico. Provee un flujo de bits que opera a una tasa de transmisión bien definida; por ejemplo, 2.4 Gbps para un enlace OC-48. El flujo de bits está organizado en forma de cargas útiles de bytes de tamaño fijo que se repiten cada 125 μseg, haya o no datos de usuario para enviar.

Para transportar paquetes a través de estos enlaces, se necesita cierto mecanismo de entramado para diferenciar los paquetes ocasionales del flujo de bits continuo en el que se transportan. PPP se ejecuta en enrutadores IP para proveer este mecanismo, como se muestra en la figura 3-23.

PPP constituye una mejora del protocolo más simple conocido como **SLIP** (**Protocolo de Línea Serial de Internet**, del inglés *Serial Line Internet Protocol*); se utiliza para manejar la configuración de detección de errores en los enlaces, soporta múltiples protocolos, permite la autentificación y tiene muchas otras funciones. Con un amplio conjunto de opciones, PPP provee tres características principales:

1. Un método de entramado que delinea sin ambigüedades el final de una trama y el inicio de la siguiente. El formato de trama también maneja la detección de errores.
2. Un protocolo de control de enlace para activar líneas, probarlas, negociar opciones y desactivarlas en forma ordenada cuando ya no son necesarias. Este protocolo se llama **LCP** (**Protocolo de Control de Enlace**, del inglés *Link Control Protocol*).

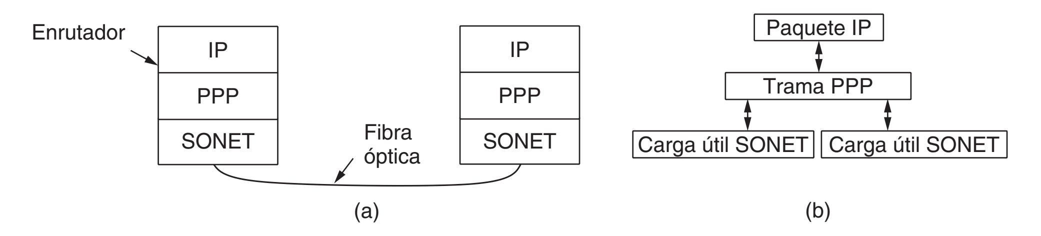


Figura 3-23. Paquete sobre SONET. (a) Una pila de protocolos. (b) Relaciones entre las tramas.

3. Un mecanismo para negociar opciones de capa de red con independencia del protocolo de red que se vaya a utilizar. El método elegido debe tener un **NCP** (**Protocolo de Control de Red**, del inglés *Network Control Protocol*) distinto para cada capa de red soportada.

El formato de trama de PPP se escogió de modo que fuera muy parecido al de **HDLC** (**Control de Enlace de Datos de Alto Nivel**, del inglés *High-level Data Link Control*), una instancia muy utilizada de una familia anterior de protocolos, ya que no había razón para reinventar la rueda.

La diferencia principal entre PPP y HDLC es que el primero está orientado a bytes, no a bits. En particular, PPP usa el relleno de bytes en las líneas y todas las tramas tienen un número entero de bytes. HDLC utiliza relleno de bytes y permite tramas de, por ejemplo, 30.25 bytes.

Sin embargo, hay una segunda diferencia importante en la práctica. HDLC provee una transmisión confiable con una ventana deslizante, confirmaciones de recepción y expiración de temporizadores en la forma que hemos visto. PPP también puede proveer una transmisión confiable en entornos ruidosos, como las redes inalámbricas; los detalles exactos se definen en el RFC 1663. Pero esto se hace pocas veces en la práctica. En vez de ello se utiliza casi siempre en Internet un “modo no numerado” para proveer un servicio sin conexión ni confirmación de recepción.

El formato de trama de PPP se muestra en la figura 3-24. Todas las tramas PPP comienzan con el byte bandera del estándar de HDLC 0x7E (01111110). Este byte de bandera se rellena con bytes si ocurre dentro del campo de carga útil (*Payload*), mediante el byte de escape 0x7D. El siguiente byte es el resultado de aplicar un XOR al byte de escape y a 0x20, con lo cual se voltea el quinto bit. Por ejemplo, 0x7D 0x5E es la secuencia de escape para el byte bandera 0x7E. Esto significa que se puede buscar el inicio y el final de las tramas con sólo explorar en busca del byte 0x7E, ya que no ocurrirá en ningún otro lado. La regla para quitar el relleno de bytes al recibir una trama es buscar el byte 0x7D, eliminarlo y aplicar un XOR al siguiente byte junto con 0x20. Además, sólo se necesita un byte bandera entre trama y trama. Se pueden usar varios bytes bandera para llenar el enlace cuando no haya tramas para enviar.

Después del byte bandera de inicio de trama sigue el campo *Dirección*. Este campo siempre se establece al valor binario 11111111 para indicar que todas las estaciones deben aceptar la trama. Al usar este valor evitamos tener que asignar direcciones de la capa de enlace de datos.

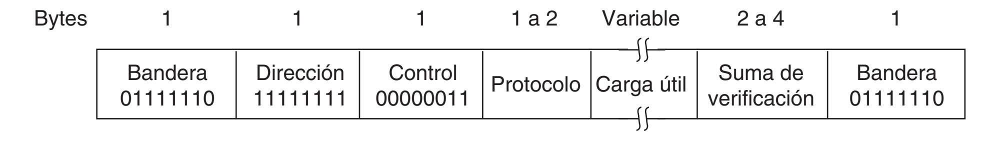


Figura 3-24. El formato de trama completa de PPP para la operación en modo no numerado.

Al campo de *Dirección* le sigue el campo de *Control*, cuyo valor predeterminado es 00000011. Este valor indica una trama no numerada.

Dado que los campos *Dirección* y *Control* son siempre constantes en la configuración predeterminada, LCP proporciona los mecanismos necesarios para que las dos partes negocien una opción que los omita por completo y ahorre 2 bytes por trama.

El cuarto campo de PPP es *Protocolo*. Su tarea es indicar la clase de paquete que está en el campo *Carga útil*. Se definen códigos que empiecen con un bit 0 para la versión 4 de IP, la versión 6 de IP y otros protocolos de capa de red que se podrían usar, como IPX y AppleTalk. Los códigos que comienzan con un bit 1 se utilizan para los protocolos de configuración de PPP, entre los cuales están LCP y un NCP diferente para cada protocolo de capa de red soportado. El tamaño predeterminado del campo *Protocolo* es de 2 bytes, pero se puede negociar a 1 byte mediante el uso de LCP. Tal vez los diseñadores fueron demasiado precavidos al pensar que algún día podría haber más de 256 protocolos en uso.

El campo *Carga útil* es de longitud variable, hasta cierto máximo negociado. Si la longitud no se negocia mediante LCP durante el establecimiento de la línea, se usa una longitud predeterminada de 1500 bytes. De ser necesario se puede incluir un relleno después de la carga útil.

Después del campo *Carga útil* está el campo *Suma de verificación*, que por lo general es de 2 bytes, aunque se puede negociar una suma de verificación de 4 bytes. La suma de verificación es, de hecho, la misma CRC de 32 bits cuyo polinomio generador se proporciona al final de la sección 3.2.2. La suma de verificación de 2 bytes es también una CRC estándar en la industria.

PPP es un mecanismo de entramado que puede transportar los paquetes de varios protocolos a través de muchos tipos de capas físicas. Para usar PPP sobre SONET, las elecciones a realizar se describen en el RFC 2615 (Malis y Simpson, 1999). Se utiliza una suma de verificación de 4 bytes, ya que éste es el medio principal para detectar errores de transmisión a través de las capas física, de enlace y de red. Se recomienda no comprimir los campos *Dirección*, *Control* y *Protocolo*, ya que los enlaces SONET operan de antemano a tasas de transmisión relativamente altas.

También hay una característica inusual. La carga útil de PPP se mezcla aleatoriamente (*scrambled*) (según lo descrito en la sección 2.5.1) antes de insertarla en la carga útil de SONET. La aleatorización aplica operaciones XOR a la carga útil con una secuencia pseudoaleatoria larga antes de transmitirla. La cuestión es que el flujo de bits de SONET necesita transiciones frecuentes de bits para la sincronización. Estas transiciones se dan en forma natural con la variación en las señales de voz, pero en la comunicación de datos el usuario selecciona la información que se envía, por lo que podría enviar un paquete con una larga secuencia de ceros. Con la aleatorización, la probabilidad de que un usuario pueda ocasionar problemas al enviar una larga secuencia de ceros es muy baja.

Antes de transportar las tramas PPP a través de líneas SONET, hay que establecer y configurar el enlace PPP. Las fases por las que pasa el enlace al activarlo, utilizarlo y desactivarlo otra vez se muestran en la figura 3-25.

El enlace inicia en el estado *MUERTO*, lo que significa que no hay conexión en la capa física. Al establecer una conexión en la capa física, el enlace pasa a *ESTABLECER*. En ese punto, los iguales de PPP intercambian una serie de paquetes LCP (cada uno de los cuales se transporta en el campo *Carga útil* de una trama PPP) para seleccionar de las posibilidades antes mencionadas, las opciones de PPP para el enlace. El igual que inicia propone las opciones y el igual que responde las acepta o las rechaza, todas o una parte de ellas. El que responde también puede hacer propuestas alternativas.

Si la negociación de opciones LCP tiene éxito, el enlace llega al estado *AUTENTIFICAR*. Ahora las dos partes pueden verificar las identidades una de la otra, si lo desean. Si la autentificación tiene éxito, el enlace entra al estado *RED* y se envía una serie de paquetes NCP para configurar la capa de red. Es difícil generalizar sobre los protocolos NCP, ya que cada uno es específico para cierto protocolo de capa de red y permite hacer peticiones de configuración específicas para ese protocolo. Por ejemplo, para IP la posibilidad más importante es la asignación de direcciones IP a ambos extremos del enlace.

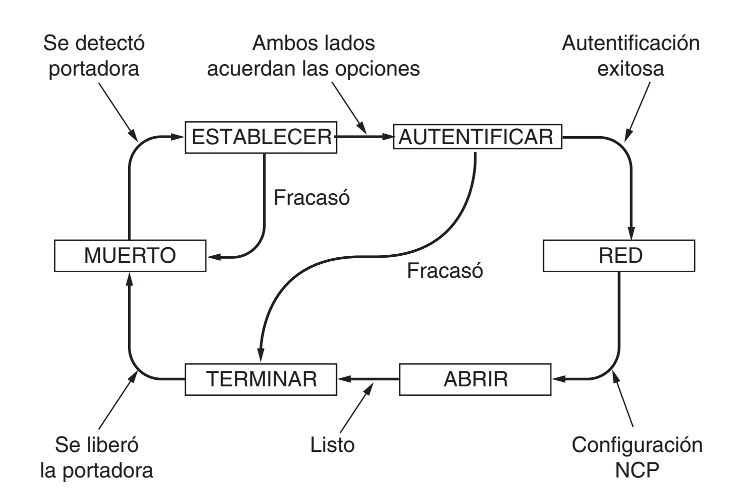


Figura 3-25. Diagrama de estado para activar y desactivar un enlace PPP.

Una vez que el enlace llega a *ABRIR*, se puede llevar a cabo el transporte de datos. En este estado es en donde se transportan los paquetes IP en tramas PPP a través de la línea SONET. Al terminar el transporte de los datos, el enlace pasa al estado *TERMINAR* y de ahí se regresa al estado *MUERTO* cuando se desactiva la conexión de la capa física.

3.5.2 ADSL

ADSL (**Línea Asimétrica de Suscriptor Digital**, del inglés *Asymmetric Digital Subscriber Loop*) conecta a millones de suscriptores desde su hogar a Internet, a tasas de transmisión de varios megabits/seg sobre el mismo lazo local telefónico que se utiliza para el servicio telefónico tradicional. En la sección 2.5.3 describimos cómo se agrega un dispositivo conocido como módem DSL al extremo del hogar. Este dispositivo envía bits sobre el lazo local a un dispositivo llamado **DSLAM** (**Multiplexor de Acceso a la ADSL**, del inglés *DSL Access Multiplexer*), el cual se encuentra en la oficina local de la compañía telefónica. Ahora exploraremos con más detalle cómo se transportan los paquetes a través de los enlaces ADSL.

En la figura 3-26 se muestra un panorama general de los protocolos y dispositivos que se utilizan con ADSL. Como se implementan distintos protocolos en diferentes redes, hemos optado por mostrar el escenario más popular. Dentro del hogar, una computadora como una PC envía paquetes IP al módem DSL mediante el uso de una capa de enlace como Ethernet. Después, el módem DSL envía los paquetes IP sobre el lazo local al DSLAM, para lo cual usa los protocolos que estudiaremos a continuación. En el DSLAM (o en un enrutador conectado a éste, dependiendo de la implementación) se extraen los paquetes IP y se introducen en una red de ISP para llegar a cualquier destino en Internet.

Los protocolos que se muestran sobre el enlace ADSL en la figura 3-26 empiezan desde la parte inferior, con la capa física de ADSL. Se basan en un esquema de modulación conocido como multiplexado por división de frecuencia ortogonal (también conocido como multitono discreto), como vimos en la sección 2.5.3. Cerca de la parte superior de la pila se encuentra PPP, justo debajo de la capa de red IP. Este protocolo es el mismo PPP que estudiamos para los transportes de paquetes sobre SONET. Funciona de la misma manera para establecer y configurar el enlace y transportar los paquetes IP.

Los protocolos ATM y AAL5 están entre ADSL y PPP. Éstos son nuevos protocolos que no hemos visto antes. El protocolo **ATM** (**Modo de Transferencia Asíncrona**, del inglés *Asynchonous Transfer*

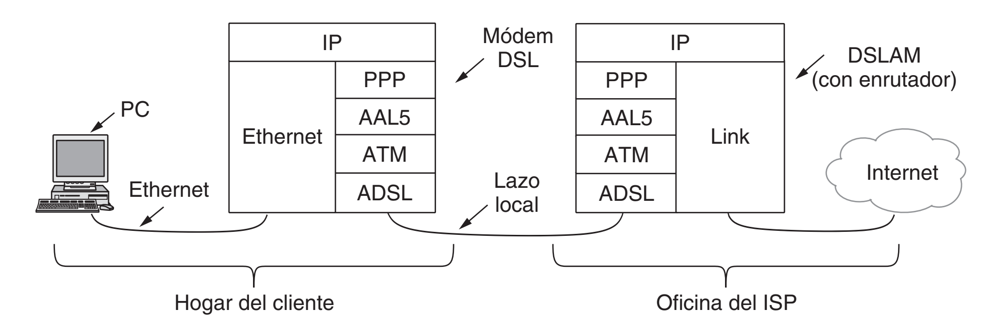


Figura 3-26. Pilas de protocolos de ADSL.

Mode) se diseñó a principios de 1990 y se lanzó con un gran despliegue publicitario. Prometía una tecnología de red que resolvería los problemas mundiales de telecomunicaciones, al fusionar la voz, los datos, la televisión por cable, el telégrafo, las palomas mensajeras, las latas conectadas mediante cordones, los tambores y todo lo demás en un sistema integrado que pudiera hacer todo para todos. Pero esto no ocurrió. En gran parte, los problemas del ATM fueron similares a los que describimos con relación a los protocolos OSI; es decir, mala sincronización, tecnología, implementación y política. Sin embargo, ATM fue mucho más exitoso que OSI. Aunque no ha conquistado el mundo, se sigue utilizando mucho en algunos nichos, incluyendo las líneas de acceso de banda ancha como DSL y los enlaces WAN dentro de las redes telefónicas.

ATM es una capa de enlace basada en la transmisión de **celdas** de información de longitud fija. Lo "asíncrono" en su nombre significa que las celdas no siempre se tienen que enviar de la misma manera en que se hace a través de las líneas sincrónicas, como en SONET. Las celdas sólo necesitan enviarse cuando haya información para transportar. ATM es una tecnología orientada a conexión. Cada celda transporta un identificador de **circuito virtual** en su encabezado y los dispositivos usan este identificador para reenviar celdas a través de las trayectorias de conexiones establecidas.

Cada una de las celdas es de 53 bytes de longitud, en donde 48 bytes son de la carga útil y 5 bytes constituyen el encabezado. Mediante el uso de celdas pequeñas, ATM puede dividir de una manera flexible el ancho de banda de un enlace de capa física entre distintos usuarios, en pequeñas porciones. Esta habilidad es útil cuando, por ejemplo, se envía tanto voz como datos a través de un enlace sin tener paquetes de datos extensos que producirían grandes variaciones en el retardo de las muestras de voz. La elección inusual en cuanto a la longitud de la celda (en comparación con la elección más natural de una potencia de 2) es una indicación de cuánta influencia política tuvo el diseño de ATM. El tamaño de 48 bytes para la carga útil fue un compromiso para resolver un interbloqueo entre Europa, que deseaba celdas de 32 bytes, y Estados Unidos, que quería celdas de 64 bytes. Si desea conocer las generalidades sobre ATM, consulte a Siu y Jain (1995).

Para enviar datos a través de una red ATM, es necesario asignarlos a una secuencia de celdas. Esta asignación se realiza mediante una capa de adaptación del ATM en un proceso llamado **segmentación** y reensamblaje. Se han definido varias capas de adaptación para distintos servicios, que varían desde los muestreos periódicos de voz hasta los datos de paquetes. La capa principal que se utiliza para los datos de paquetes es **AAL5** (**Capa de Adaptación de ATM 5**, del inglés *ATM Adaptation Layer 5*).

En la figura 3-27 se muestra una trama AAL5. En vez de encabezado, tiene un terminador que proporciona la longitud y cuenta con una CRC de 4 bytes para la detección de errores. Naturalmente, la CRC es la misma que se utiliza para las redes PPP y LAN IEEE 802 como Ethernet. Wang y Crowcroft (1992) demostraron que es lo bastante sólida como para detectar errores no tradicionales como el reordenamiento de las celdas. Al igual que una carga útil, la trama AAL5 tiene relleno. Esto redondea la longitud total para

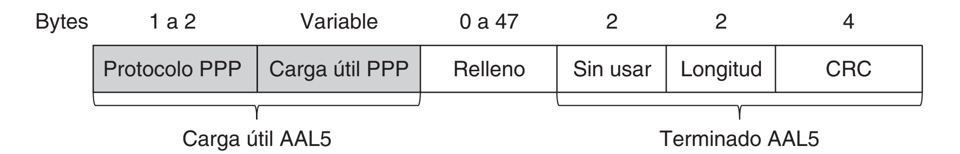


Figura 3-27. Trama AAL5 que transporta datos PPP.

que sea un múltiplo de 48 bytes, de modo que la trama se pueda dividir equitativamente en celdas. No se necesitan direcciones en la trama, ya que el identificador de circuito virtual incluido en cada celda la llevará al destino correcto.

Ahora que hemos descrito el ATM, sólo nos queda describir cómo es que PPP usa a ATM en el caso de ADSL. Esto se hace con otro estándar llamado **PPPoA** (**PPP sobre ATM**, del inglés *PPP over ATM*). En realidad este estándar no es un protocolo (por lo que no aparece en la figura 3-26), sino más bien una especificación sobre cómo trabajar con tramas PPP y AAL5. Se describe en el RFC 2364 (Gross y colaboradores, 1998).

Sólo los campos de protocolo y carga útil de PPP se colocan en la carga útil de AAL5, como se muestra en la figura 3-27. El campo de protocolo indica al DSLAM en el extremo lejano si la carga útil es un paquete IP o un paquete de otro protocolo tal como LCP. El extremo lejano sabe que las celdas contienen información PPP, ya que se estableció un circuito virtual ATM para este fin.

Dentro de la trama AAL5 no se necesita entramado PPP, ya que no serviría para ningún propósito puesto que ATM y AAL5 proveen de antemano el entramado. Más entramado sería inútil. Tampoco se necesita la CRC de PPP, ya que AAL5 incluye la misma CRC. Este mecanismo de detección de errores complementa la codificación de capa física de ADSL que consta de un código de Reed-Solomon para corrección de errores y una CRC de 1 byte para la detección de los errores restantes que no se hayan detectado. Este esquema tiene un mecanismo de recuperación de errores mucho más sofisticado que cuando se envían paquetes a través de una línea SONET, ya que ADSL es un canal mucho más ruidoso.

3.6 RESUMEN

La tarea de la capa de enlace de datos es convertir el flujo de bits en bruto ofrecido por la capa física en un flujo de tramas para que la capa de red lo utilice. La capa de enlace puede presentar este flujo con niveles variables de confiabilidad, desde un servicio sin conexión ni confirmación de recepción hasta un servicio confiable, orientado a conexión.

Se emplean varios métodos de entramado, incluidos el conteo de bytes, el relleno de bytes y el relleno de bits. Los protocolos de enlace de datos pueden proporcionar control de errores para detectar o corregir tramas dañadas y retransmitir tramas perdidas. Para evitar que un emisor rápido sature a un receptor lento, el protocolo de enlace de datos también puede proveer control de flujo. El mecanismo de ventana deslizante se emplea ampliamente para integrar el control de errores y el control de flujo de una manera simple. Cuando el tamaño de la ventana es de un paquete, el protocolo es de parada y espera.

Los códigos para corrección y detección de errores agregan información redundante a los mensajes mediante el uso de una variedad de técnicas matemáticas. Los códigos convolucionales y los códigos de Reed-Solomon se emplean mucho para corrección de errores; la popularidad de los códigos de verificación de paridad de baja densidad va en aumento. Los códigos para detección de errores que se utilizan en la práctica incluyen las comprobaciones de redundancias cíclicas y las sumas de verificación. Todos estos códigos se pueden aplicar en la capa de enlace datos, así como en la capa física y en capas superiores.

En este capítulo examinamos una serie de protocolos que proveen una capa de enlace confiable mediante el uso de confirmaciones de recepción y retransmisiones, o **ARQ** (**Solicitud Automática de Repetición**, del inglés *Automatic Repeat reQuest*), bajo consideraciones más realistas. Partiendo de un entorno libre de errores en donde el receptor puede manejar cualquier trama que se le envíe, introdujimos el control de flujo, seguido del control de errores con números de secuencia y el algoritmo de parada y espera. Después usamos el algoritmo de ventana deslizante para permitir una comunicación bidireccional, y presentamos el concepto de superposición (*piggybacking*). Los últimos dos protocolos canalizan la transmisión de múltiples tramas para evitar que el emisor bloquee un enlace con un retardo de propagación largo. El receptor puede desechar todas las tramas que no sean la siguiente en secuencia, o colocar en el búfer las tramas fuera de secuencia y enviar confirmaciones de recepción negativas para una mayor eficiencia en el ancho de banda. La primera estrategia es un protocolo de retroceso n, y la segunda es un protocolo de repetición selectiva.

Internet utiliza PPP como el protocolo de enlace de datos principal sobre líneas punto a punto. Provee un servicio sin conexión ni confirmación de recepción mediante el uso de bytes bandera para delimitar las tramas, junto con una CRC para detección de errores. Se utiliza para transportar paquetes a través de un intervalo de enlaces, incluyendo enlaces SONET en redes de área amplia y enlaces ADSL para el hogar.

PROBLEMAS

1. Un paquete de capa superior se divide en 10 tramas, cada una de las cuales tiene un 80% de probabilidad de llegar sin daño. Si el protocolo de enlace de datos no lleva a cabo el control de errores, ¿cuántas veces se debe reenviar el mensaje en promedio para conseguir que pase todo?

2. La siguiente codificación de caracteres se utiliza en un protocolo de enlace de datos:

A: 01000111 B: 11100011 FLAG: 01111110 ESC: 11100000

Muestre la secuencia de bits transmitida (en binario) para la trama de cuatro caracteres: A B ESC FLAG cuando se utiliza cada uno de los siguientes métodos de entramado:
(a) Conteo de caracteres.
(b) Bytes bandera con relleno de bytes.
(c) Bytes bandera de inicio y final, con relleno de bits.

3. El siguiente fragmento de datos ocurre a la mitad de un flujo de datos para el cual se utiliza el algoritmo de relleno de bytes descrito en el texto: A B ESC C ESC FLAG FLAG D. ¿Cuál es la salida después del relleno?

4. ¿Cuál es la sobrecarga máxima en el algoritmo de relleno de bytes?

5. Uno de sus compañeros, de nombre Scrooge, ha señalado que es un desperdicio terminar cada trama con un byte bandera e iniciar la siguiente con otro. Un solo byte bandera podría hacer el trabajo, por lo que un byte guardado es un byte ganado. ¿Está usted de acuerdo?

6. Una cadena de bits, 0111101111101111110, necesita transmitirse en la capa de enlace de datos. ¿Cuál es la cadena que realmente se transmite después del relleno de bits?

7. ¿Puede pensar en alguna circunstancia en la que podría ser preferible un protocolo de lazo abierto (por ejemplo, un código de Hamming) a los protocolos tipo retroalimentación que vimos en este capítulo?

8. Para proporcionar mayor confiabilidad de la que puede dar un solo bit de paridad, un esquema de codificación de detección de errores usa un bit de paridad para verificar todos los bits de número impar y un segundo bit de paridad para todos los bits de número par. ¿Cuál es la distancia de Hamming de este código?

9. Se van a transmitir mensajes de 16 bits mediante un código de Hamming. ¿Cuántos bits de verificación se necesitan para asegurar que el receptor pueda detectar y corregir errores de un solo bit? Muestre el patrón de bits transmitido para el mensaje 1101001100110101. Suponga que se utiliza paridad par en el código de Hamming.

10. Un código de Hamming de 12 bits, cuyo valor hexadecimal es 0xE4F, llega al receptor. ¿Cuál era el valor original en hexadecimal? Suponga que a lo más hay 1 bit con error.

11. Una manera de detectar errores es transmitir los datos como un bloque de *n* filas de *k* bits por fila y agregar bits de paridad a cada fila y a cada columna. El bit en la esquina inferior derecha es un bit de paridad que verifica su fila y su columna. ¿Detectará este esquema todos los errores sencillos? ¿Los errores dobles? ¿Los errores triples? Muestre que este esquema no puede detectar algunos errores de cuatro bits.

12. Suponga que se transmiten datos en bloques con tamaños de 1000 bits. ¿Cuál es la máxima tasa de error bajo la cual es mejor usar el mecanismo de detección de errores y retransmisión (1 bit de paridad por bloque) que el código de Hamming? Suponga que los errores de bits son independientes unos de otros y que no hay error de bit durante la retransmisión.

13. Un bloque de bits con *n* filas y *k* columnas usa bits de paridad horizontales y verticales para la detección de errores. Suponga que se invierten exactamente 4 bits debido a errores de transmisión. Deduzca una expresión para la probabilidad de que no se detecte el error.

14. Si utiliza el codificador convolucional de la figura 3-7, ¿cuál es la secuencia de salida cuando la secuencia de entrada es 10101010 (de izquierda a derecha) y en el estado interno inicial todos los bits son cero?

15. Suponga que se transmite un mensaje 1001 1100 1010 0011 mediante el uso de la suma de verificación de Internet (palabra de 4 bits). ¿Cuál es el valor de la suma de verificación?

16. ¿Qué residuo se obtiene al dividir $x^7 + x^5 + 1$ entre el polinomio generador $x^3 + 1$?

17. Un flujo de bits 10011101 se transmite utilizando el método estándar CRC que se describió en el texto. El generador polinomial es $x^3 + 1$. Muestre la cadena de bits real que se transmite. Suponga que el tercer bit, de izquierda a derecha, se invierte durante la transmisión. Muestre que este error se detecta en el lado receptor. Mencione un ejemplo de errores de bits en la cadena de bits transmitida que no serán detectados por el receptor.

18. Se envía un mensaje de 1024 bits que contiene 992 bits de datos y 32 bits de CRC. La CRC se calcula mediante el polinomio CRC de 32 grados estandarizado de IEEE 802. Para cada uno de los siguientes casos, explique si el receptor detectará los errores durante la transmisión del mensaje:
(a) Hubo un error de un solo bit.
(b) Hubo dos errores de bit aislados.
(c) Hubo 18 errores de bit aislados.
(d) Hubo 47 errores de bit aislados.
(e) Hubo un error de ráfaga extenso de 24 bits.
(f) Hubo un error de ráfaga extenso de 35 bits.

19. En el análisis del protocolo ARQ en la sección 3.3.3, se describió un escenario en el que el receptor aceptaba dos copias de la misma trama debido a la pérdida de la trama de confirmación de recepción. ¿Es posible que un receptor pueda aceptar múltiples copias de la misma trama si no se pierde ninguna de las tramas (mensaje o confirmación de recepción)?

20. Un canal tiene una tasa de bits de 4 kbps y un retardo de propagación de 20 mseg. ¿Para qué intervalo de tamaños de trama se obtiene una eficiencia de cuando menos 50% con el protocolo de parada y espera?

21. En el protocolo 3, ¿es posible que el emisor inicie el temporizador cuando éste ya está en ejecución? De ser así, ¿cómo podría ocurrir? De lo contrario, ¿por qué no es posible?

22. Una troncal T1 de 3 000 km de longitud se usa para transmitir tramas de 64 bytes con el protocolo 5. Si la velocidad de propagación es de 6 μseg/km, ¿de cuántos bits deben ser los números de secuencia?

23. Imagine un protocolo de ventana deslizante que utiliza tantos bits para los números de secuencia que nunca ocurre un reinicio. ¿Qué relaciones deben mantenerse entre los cuatro límites de la ventana y el tamaño de la misma, que es constante y el mismo tanto para el emisor como para el receptor?

24. Si el procedimiento *between* del protocolo 5 verificara la condición $a \leq b \leq c$ en lugar de la condición $a \leq b \leq c$, ¿tendría esto algún efecto en la corrección o en la eficiencia del protocolo? Explique su respuesta.

25. En el protocolo 6, cuando llega una trama de datos, se hace una revisión para ver si el número de secuencia es diferente del esperado y si el valor de *no_nak* es verdadero. Si ambas condiciones se cumplen, se envía una

NAK. De otra manera, se inicia el temporizador auxiliar. Suponga que se omite la cláusula else. ¿Afectará este cambio la corrección del protocolo?

26. Suponga que el ciclo while de tres instrucciones cerca del final del protocolo 6 se elimina del código. ¿Afectará esto la corrección del protocolo o sólo su desempeño? Explique su respuesta.

27. La distancia desde la Tierra a un planeta distante es de aproximadamente 9×10^9 m. ¿Cuál es la utilización del canal si se usa un protocolo de parada y espera para la transmisión de tramas en un enlace punto a punto de 64 Mbps? Suponga que el tamaño de trama es de 32 KB y que la velocidad de la luz es de 3×10^8 m/s.

28. En el problema anterior, suponga que ahora se utiliza un protocolo de ventana deslizante. Para qué tamaño de ventana de emisor la utilización del enlace será de 100%? Puede ignorar los tiempos de procesamiento del protocolo en el emisor y el receptor.

29. En el protocolo 6, el código de *frame_arrival* tiene una sección que se usa para las NAK. Dicha sección se invoca si la trama entrante es una NAK y se cumple otra condición. Describa un escenario en el que la presencia de esta otra condición sea esencial.

30. Considere la operación del protocolo 6 en una línea de 1 Mbps perfecta (es decir, libre de errores). El tamaño máximo de trama es de 1000 bits. Se generan nuevos paquetes a intervalos aproximados de 1 segundo. El tiempo de expiración del temporizador es de 10 mseg. Si se eliminara el temporizador especial de confirmación de recepción, ocurrirían terminaciones de temporizador innecesarias. ¿Cuántas veces se transmitiría el mensaje promedio?

31. En el protocolo 6, $MAX_SEQ = 2^n - 1$. Si bien esta condición es evidentemente deseable para utilizar de manera eficiente los bits de encabezado, no hemos demostrado que sea esencial. ¿Funciona correctamente el protocolo con $MAX_SEQ = 4$, por ejemplo?

32. Se están enviando tramas de 1000 bits a través de un canal de 1 Mbps mediante el uso de un satélite geoestacionario, cuyo tiempo de propagación desde la Tierra es de 270 mseg. Las confirmaciones de recepción siempre se superponen en las tramas de datos. Los encabezados son muy cortos. Se usan números de secuencia de tres bits. ¿Cuál es la utilización máxima de canal que se puede lograr para:
(a) Parada y espera?
(b) El protocolo 5?
(c) El protocolo 6?

33. Calcule la fracción del ancho de banda que se desperdicia en sobrecarga (encabezados y retransmisiones) para el protocolo 6 en un canal satelital de 50 kbps con carga pesada, en donde se utilicen tramas de datos consistentes en 40 bits de encabezado y 3960 bits de datos. Asuma que el tiempo de propagación de la señal de la Tierra al satélite es de 270 mseg. Nunca ocurren tramas ACK. Las tramas NAK son de 40 bits. La tasa de error de las tramas de datos es de 1%, y la tasa de error para las tramas NAK es insignificante. Los números de secuencia son de 8 bits.

34. Considere un canal satelital de 64 kbps libre de errores que se usa para enviar tramas de datos de 512 bytes en una dirección y devolver confirmaciones de recepción muy cortas en la otra. ¿Cuál es la velocidad real de transmisión máxima para tamaños de ventana de 1, 7, 15 y 127? El tiempo de propagación de la Tierra al satélite es de 270 mseg.

35. Un cable de 100 km de longitud opera con una tasa de datos de línea T1. La velocidad de propagación del cable es 2/3 de la velocidad de la luz en el vacío. ¿Cuántos bits caben en el cable?

36. Dé por lo menos una razón por la cual PPP utiliza relleno de bytes en vez de relleno de bits para evitar que bytes bandera accidentales dentro de la carga útil causen confusión.

37. ¿Cuál es la sobrecarga mínima para enviar un paquete IP mediante PPP? Tome en cuenta sólo la sobrecarga introducida por el mismo PPP, no la del encabezado IP. ¿Cuál es la sobrecarga máxima?

38. Un paquete IP de 100 bytes se transmite sobre un lazo local mediante el uso de una pila de protocolos de ADSL. ¿Cuántas celdas ATM se transmitirán? Describa brevemente su contenido.

39. El objetivo de este ejercicio de laboratorio es implementar un mecanismo de detección de errores mediante el algoritmo CRC estándar descrito en el texto. Escriba dos programas: el *generador* y el *verificador*. El progra-

ma *generador* lee de la entrada estándar una línea de texto ASCII que contiene un mensaje de n bits, el cual consiste en una cadena de 0s y 1s. La segunda línea es el polinomio de k bits, también en ASCII. Ésta envía a la salida estándar una línea de texto ASCII con $n + k$ 0s y 1s que representan el mensaje a transmitir. Después envía el código polinomial, justo como lo lee. El programa verificador lee la salida del programa generador y envía a la salida un mensaje que indica si es correcto o no. Por último, escriba un programa llamado *alterar* que invierta 1 bit en la primera línea dependiendo de su argumento (el número de bits, considerando el bit de más a la izquierda como 1), pero que copie el resto de las dos líneas de manera correcta. Si escribe:

```
generador <archivo | verificador
```

deberá ver que el mensaje es correcto, pero si escribe

```
generador <archivo | alterar arg | verificador
```

deberá obtener el mensaje de error.

4

LA SUBCAPA DE CONTROL DE ACCESO AL MEDIO

Los enlaces de red se pueden dividir en dos categorías: los que utilizan conexiones punto a punto y los que utilizan canales de difusión. En el capítulo 2 estudiamos los enlaces punto a punto; este capítulo trata sobre las redes de difusión y sus protocolos.

En cualquier red de difusión, el asunto clave es la manera de determinar quién puede utilizar el canal cuando tiene competencia por él. Para aclarar este punto, considere una llamada en conferencia en la que seis personas, en seis teléfonos diferentes, están conectadas de modo que cada una puede oír y hablar con todas las demás. Es muy probable que cuando una de ellas deje de hablar, dos o más comiencen a hacerlo al mismo tiempo, lo que conducirá al caos. En una reunión cara a cara, el caos se evita por medios competentes. Por ejemplo, en una reunión la gente levanta la mano para solicitar permiso para hablar. Cuando sólo hay un canal disponible, es mucho más difícil determinar quién debería tener el turno. Se conocen muchos protocolos para resolver el problema, los cuales forman el contenido de este capítulo. En la literatura, los canales de difusión a veces se denominan **canales multiacceso** o **canales de acceso aleatorio**.

Los protocolos que se utilizan para determinar quién sigue en un canal multiacceso pertenecen a una subcapa de la capa de enlace de datos llamada subcapa **MAC** (**Control de Acceso al Medio**, del inglés *Medium Access Control*). La subcapa MAC tiene especial importancia en las LAN, en especial las inalámbricas puesto que el canal inalámbrico es de difusión por naturaleza. En contraste, las WAN usan enlaces punto a punto, excepto en las redes satelitales. Debido a que los canales multiacceso y las LAN están muy relacionados, en este capítulo analizaremos las LAN en general, además de algunos aspectos que no son estrictamente parte de la subcapa MAC, pero el tema principal será el control del canal.

Desde el punto de vista técnico, la subcapa MAC es la parte inferior de la capa de enlace de datos, por lo que lógicamente debimos haberla estudiado antes de examinar los protocolos punto a punto en el capítulo 3. No obstante, para la mayoría de la gente, la comprensión de los protocolos en los que intervienen muchas partes es más fácil una vez que se han entendido

bien los protocolos de dos partes. Por esta razón nos hemos desviado un poco de un orden de presentación estrictamente ascendente.

4.1 EL PROBLEMA DE ASIGNACIÓN DEL CANAL

El tema central de este capítulo es la forma de asignar un solo canal de difusión entre usuarios competidores. El canal podría ser una parte del espectro inalámbrico en una región geográfica, o un solo alambre o fibra óptica en donde se conectan varios nodos. Esto no es importante. En ambos casos, el canal conecta a cada usuario con todos los demás; cualquier usuario que utilice todo el canal interfiere con los demás que también desean usarlo.

Primero veremos las deficiencias de los esquemas de asignación estática para el tráfico en ráfagas. Después estableceremos las suposiciones clave que se utilizan para modelar los esquemas dinámicos que examinaremos en las siguientes secciones.

4.1.1 Asignación estática de canal

La manera tradicional de asignar un solo canal, como una troncal telefónica, entre múltiples usuarios competidores es dividir su capacidad mediante el uso de uno de los esquemas de multiplexión que describimos en la sección 2.5, como el **FDM** (**Multiplexión por División de Frecuencia**, del inglés *Frequency Division Multiplexing*). Si hay *N* usuarios, el ancho de banda se divide en *N* partes de igual tamaño, y a cada usuario se le asigna una parte. Debido a que cada usuario tiene una banda de frecuencia privada, ahora no hay interferencia entre ellos. Cuando sólo hay una pequeña cantidad fija y constante de usuarios, cada uno tiene un flujo estable o una carga de tráfico pesada, esta división es un mecanismo de asignación sencillo y eficiente. Las estaciones de radio de FM son un ejemplo inalámbrico. Cada estación recibe una parte de la banda de FM y la utiliza la mayor parte del tiempo para difundir su señal.

Sin embargo, cuando el número de emisores es grande y varía continuamente, o cuando el tráfico se hace en ráfagas, el FDM presenta algunos problemas. Si el espectro se divide en *N* regiones y actualmente hay menos de *N* usuarios interesados en comunicarse, se desperdiciará una buena parte del valioso espectro. Y si más de *N* usuarios quieren comunicarse, a algunos de ellos se les negará el permiso por falta de ancho de banda, aun cuando algunos de los usuarios que tengan asignada una banda de frecuencia apenas transmitan o reciban algo.

Aun suponiendo que el número de usuarios podría, de alguna manera, mantenerse constante en *N*, dividir el único canal disponible en varios subcanales estáticos es ineficiente por naturaleza. El problema básico es que, cuando algunos usuarios están inactivos, su ancho de banda simplemente se pierde. No lo están usando, y a nadie más se le permite usarlo. Una asignación estática es un mal arreglo para la mayoría de los sistemas de cómputo, en donde el tráfico de datos se presenta en ráfagas muy intensas, a menudo con relaciones de tráfico pico a tráfico medio de 1000:1. En consecuencia, la mayoría de los canales estarán inactivos casi todo el tiempo.

El pobre desempeño del FDM estático se puede ver fácilmente mediante un cálculo sencillo de la teoría de colas. Primero obtendremos el retardo promedio, T, al enviar una trama a través de un canal con C bps de capacidad. Suponemos que las tramas llegan al azar con una tasa de llegada promedio de λ tramas/seg y que la longitud de las tramas es variable, con una longitud promedio de $1/\mu$ bits. Con estos parámetros, la tasa de servicio del canal es de μC tramas/seg. Un resultado de la teoría de colas estándar es

$$T = \frac{1}{\mu C - \lambda}$$

(para los curiosos, este resultado es para una cola “M/M/1” y requiere que la aleatoriedad de los tiempos entre las llegadas de las tramas y las longitudes de éstas sigan una distribución exponencial, o que en su defecto sea el resultado de un proceso de Poisson).

En nuestro ejemplo, si C es 100 Mbps, la longitud promedio de trama, $1/\mu$, es 10 000 bits y la tasa de llegada de tramas, λ, es 5 000 tramas/seg, entonces $T = 200$ μseg. Cabe mencionar que si ignoráramos el retardo de colas y sólo preguntáramos cuánto toma enviar una trama de 10 000 bits en una red de 100 Mbps, podríamos obtener la respuesta (incorrecta) de 100 μseg. Este resultado sólo es válido cuando no hay competencia por el canal.

Ahora dividamos el canal en N subcanales independientes, cada uno con capacidad de C/N bps. La tasa promedio de entrada de cada uno de los subcanales ahora será de λ/N. Al recalcular T, obtenemos:

$$T_N = \frac{1}{\mu(C/N) - (\lambda/N)} = \frac{N}{\mu C - \lambda} = NT \quad (4\text{-}1)$$

El retardo promedio al usar el canal dividido es N veces peor que si todas las tramas se acomodaran mágicamente de algún modo en una gran cola central. Este mismo resultado nos dice que en el vestíbulo de un banco lleno de máquinas ATM es mejor tener una sola cola que alimente a todas las máquinas, que una cola separada en frente de cada máquina.

Precisamente los mismos argumentos que se aplican a la FDM se adaptan a otras formas de dividir estáticamente el canal. Si se usara la multiplexión por división de tiempo (TDM) y a cada usuario se le asignara cada N-ésima ranura de tiempo, en caso de que un usuario no utilizara la ranura asignada, simplemente se desperdicia. Lo mismo se aplica si dividimos las redes físicamente. Si utilizamos otra vez el ejemplo anterior reemplazamos la red de 100 Mbps con 10 redes de 10 Mbps cada una, asignándole de manera estática un usuario a cada una de ellas, el retardo promedio se iría de 200 μseg a 2 mseg.

Ya que ninguno de los métodos tradicionales de asignación estática de canal funciona bien con tráfico en ráfagas, ahora exploraremos los métodos dinámicos.

4.1.2 Supuestos para la asignación dinámica de canales

Antes de entrar en el primero de muchos métodos de asignación de canal que veremos en este capítulo, vale la pena formular con cuidado el problema de la asignación. Todo el trabajo hecho en esta área se basa en cinco supuestos clave, que se describen a continuación:

1. **Tráfico independiente.** El modelo consiste en N **estaciones** independientes (computadoras, teléfonos), cada una con un programa o usuario que genera tramas para transmisión. El número esperado de tramas que se generan en un intervalo de longitud Δt es de $\lambda \Delta t$, donde λ es una constante (la tasa de llegada de tramas nuevas). Una vez que se ha generado una trama, la estación se bloquea y no hace nada sino hasta que la trama se haya transmitido con éxito.
2. **Canal único.** Hay un solo canal disponible para todas las comunicaciones. Todas las estaciones pueden transmitir en él y pueden recibir de él. Se asume que las estaciones tienen una capacidad equivalente, aunque los protocolos pueden asignarles distintos roles (prioridades).
3. **Colisiones observables.** Si dos tramas se transmiten en forma simultánea, se traslapan en el tiempo y la señal resultante se altera. Este evento se llama **colisión**. Todas las estaciones pueden detectar una colisión que haya ocurrido. Una trama en colisión se debe volver a transmitir después. No hay otros errores, excepto aquéllos generados por las colisiones.
4. **Tiempo continuo o ranurado.** Se puede asumir que el tiempo es continuo, en cuyo caso la transmisión de una trama puede comenzar en cualquier momento. Por el contrario, el tiempo se puede ranurar o dividir en intervalos discretos (llamados ranuras). En este caso las transmisiones de las

tramas deben empezar al inicio de una ranura. Una ranura puede contener 0, 1 o más tramas, correspondientes a una ranura inactiva, una transmisión exitosa o una colisión, respectivamente.

5. **Detección de portadora o sin detección de portadora.** Con el supuesto de detección de portadora, las estaciones pueden saber si el canal está en uso antes de intentar usarlo. Si se detecta que el canal está ocupado, ninguna estación intentará utilizarlo. Si no hay detección de portadora, las estaciones no pueden detectar el canal antes de intentar usarlo. Simplemente transmiten. Sólo después pueden determinar si la transmisión tuvo éxito.

Es importante un análisis de estos supuestos. El primero dice que las llegadas de las tramas son independientes, tanto en todas las estaciones como en una estación específica, y que las tramas se generan en forma impredecible, pero a una tasa de transmisión constante. En realidad este supuesto no es en sí un buen modelo de tráfico de red, pues es bien sabido que los paquetes llegan en ráfagas durante un rango de escalas de tiempo (Paxson y Floyd, 1995; y Leland y colaboradores, 1994). Sin embargo, los **modelos de Poisson**, como se les dice con frecuencia, son útiles debido a que son matemáticamente de fácil solución. Estos modelos nos ayudan a analizar protocolos para comprender a grandes rasgos cómo cambia el rendimiento durante un intervalo de operación y cómo se compara con otros diseños.

El supuesto del canal único es la esencia del modelo. No existen formas externas de comunicación. Las estaciones no pueden levantar la mano para solicitar que el maestro les ceda la palabra, por lo que tendremos que idear mejores soluciones.

Los tres supuestos restantes dependen de la ingeniería del sistema; le diremos cuáles supuestos son válidos cuando examinemos un protocolo en particular.

El supuesto de colisión es básico. Las estaciones necesitan una forma de detectar las colisiones si quieren retransmitir las tramas en vez de dejar que se pierdan. En los canales alámbricos, se puede diseñar el hardware de los nodos para detectar las colisiones cuando éstas ocurran. Así, las estaciones pueden terminar sus transmisiones en forma prematura para evitar desperdiciar capacidad. Esta detección es mucho más difícil para los canales inalámbricos, por lo que las colisiones casi siempre se deducen después de que ocurren, debido a que se esperaba una trama de confirmación de recepción y nunca llegó. También es posible que se reciban algunas tramas involucradas en una colisión, dependiendo de los detalles de las señales y del hardware receptor. Pero como esta situación no es el caso común, supongamos que se pierden todas las tramas involucradas en una colisión. También veremos protocolos diseñados sobre todo para evitar que ocurran colisiones.

La razón de las dos suposiciones alternativas sobre el tiempo es que el tiempo ranurado se puede usar para mejorar el rendimiento. Sin embargo, requiere que las estaciones sigan un reloj maestro o que sincronicen sus acciones entre sí para dividir el tiempo en intervalos discretos. Por ende, no siempre está disponible. En este texto estudiaremos y analizaremos sistemas con ambos tipos de supuestos sobre el tiempo. Para un sistema dado, sólo uno de ellos es válido.

De manera similar, una red puede tener detección de portadora o no. Por lo general, las redes alámbricas tienen esta detección de portadora. Las redes inalámbricas no siempre la pueden utilizar de manera efectiva porque tal vez no todas las estaciones estén dentro del rango radial de las demás. Asimismo, la detección de portadora no estará disponible en otros entornos en donde una estación no se pueda comunicar directamente con otra estación, por ejemplo un módem de cable en el cual las estaciones deben comunicarse a través del amplificador de cabecera. Cabe mencionar que la palabra "portadora" en este sentido se refiere a una señal eléctrica en el cable y no tiene nada que ver con las portadoras comunes (por ejemplo, las compañías telefónicas), que datan de los tiempos del Pony Express.

Para evitar cualquier malentendido, debemos tener en cuenta que ningún protocolo multiacceso garantiza una entrega confiable. Aun cuando no haya colisiones, el receptor puede haber copiado alguna trama en forma incorrecta por diversas razones. Otras partes de la capa de enlace o las capas superiores se encargan de proveer confiabilidad.

4.2 PROTOCOLOS DE ACCESO MÚLTIPLE

Se conocen muchos algoritmos para asignar un canal de acceso múltiple. En las siguientes secciones estudiaremos una muestra representativa de los más interesantes y daremos algunos ejemplos de cómo se usan comúnmente en la práctica.

4.2.1 ALOHA

La historia de nuestro primer MAC empieza en la prístina Hawai de principios de la década de 1970. En este caso, podemos interpretar "prístina" como que "no tenía un sistema telefónico funcional". Esto no hizo la vida más placentera para el investigador Norman Abramson y sus colegas de la Universidad de Hawai, quienes trataban de conectar a los usuarios en islas remotas a la computadora principal en Honolulu. Tender sus propios cables bajo el Océano Pacífico no era una opción viable, por lo que buscaron una solución diferente.

La que encontraron utilizaba radios de corto rango, en donde cada terminal de usuario compartía la misma frecuencia ascendente para enviar tramas a la computadora central. Incluía un método simple y elegante para resolver el problema de asignación de canal. Desde entonces, su trabajo ha sido extendido por muchos investigadores (Schwartz y Abramson, 2009). Aunque el trabajo de Abramson, llamado sistema ALOHA, usó la radiodifusión basada en tierra, la idea básica es aplicable a cualquier sistema en el que usuarios no coordinados compiten por el uso de un solo canal compartido.

Analizaremos dos versiones de ALOHA: puro y ranurado. Difieren en cuanto a si el tiempo es continuo, como en la versión pura, o si se divide en ranuras discretas en las que deben caber todas las tramas.

ALOHA puro

La idea básica de un sistema ALOHA es sencilla: permitir que los usuarios transmitan cuando tengan datos por enviar. Por supuesto, habrá colisiones y las tramas en colisión se dañarán. Los emisores necesitan alguna forma de saber si éste es el caso. En el sistema ALOHA, después de que cada estación envía su trama a la computadora central, ésta vuelve a difundir la trama a todas las estaciones. Así, una estación emisora puede escuchar la difusión de la estación terrena maestra (*hub*) para ver si pasó su trama o no. En otros sistemas, como las LAN alámbricas, el emisor podría ser capaz de escuchar si hay colisiones mientras transmite.

Si la trama fue destruida, el emisor simplemente espera un tiempo aleatorio y la envía de nuevo. El tiempo de espera debe ser aleatorio o las mismas tramas chocarán una y otra vez, en sincronía. Los sistemas en los cuales varios usuarios comparten un canal común de modo tal que puede dar pie a conflictos se conocen como sistemas de **contención**.

En la figura 4-1 se presenta un esbozo de la generación de tramas en un sistema ALOHA. Hemos hecho que todas las tramas sean de la misma longitud porque la velocidad real de transmisión (*throughput*) de los sistemas ALOHA se maximiza al tener tramas con un tamaño uniforme en lugar de tramas de longitud variable.

Cada vez que dos tramas traten de ocupar el canal al mismo tiempo, habrá una colisión (como se muestra en la figura 4-1) y ambas se dañarán. Si el primer bit de una trama nueva se traslapa con el último bit de una trama casi terminada, ambas se destruirán por completo (es decir, tendrán sumas de verificación incorrectas) y ambas tendrán que volver a transmitirse más tarde. La suma de verificación no distingue (y no debe) entre una pérdida total y un error ligero. Lo malo es malo.

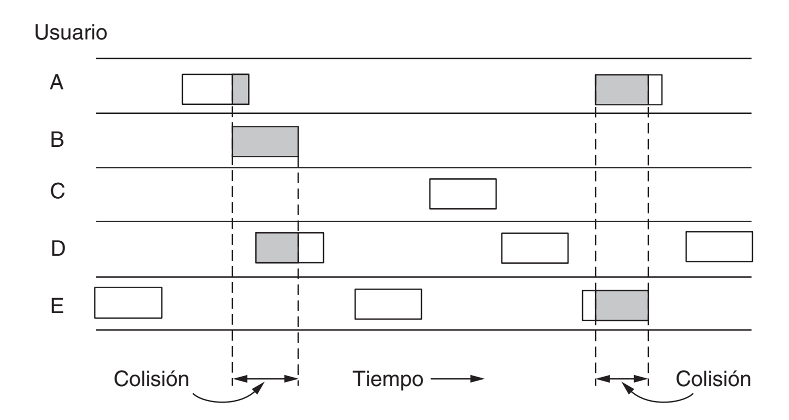


Figura 4-1. En ALOHA puro, las tramas se transmiten en tiempos completamente arbitrarios.

Hay una pregunta interesante: ¿cuál es la eficiencia de un canal ALOHA? En otras palabras, ¿qué fracción de todas las tramas transmitidas escapa a las colisiones en estas caóticas circunstancias? Primero consideremos un conjunto infinito de usuarios escribiendo en sus terminales (estaciones). Un usuario siempre está en uno de dos estados: escribiendo o esperando. Al principio todos los usuarios están en el estado de escritura. Al terminar una línea, el usuario deja de escribir, en espera de una respuesta. Después, la estación transmite una trama (que contiene la línea) a través del canal compartido hasta la computadora central y verifica el canal para saber si llegó con éxito. De ser así, el usuario ve la respuesta y continúa escribiendo. Si no, el usuario continúa esperando mientras la estación transmite la trama una y otra vez hasta que se envía con éxito.

Hagamos que el "tiempo de trama" denote el tiempo necesario para transmitir la trama estándar de longitud fija (es decir, la longitud de la trama dividida entre la tasa de bits). En este punto, suponemos que las tramas nuevas generadas por las estaciones están bien modeladas según una distribución de Poisson con una media de N tramas por tiempo de trama (la suposición de población infinita es necesaria para asegurar que N no disminuya a medida que se bloquean los usuarios). Si $N > 1$, la comunidad de usuarios está generando tramas a una tasa mayor que la que puede manejar el canal, y casi todas las tramas sufrirán una colisión. Para una velocidad de transmisión razonable esperaríamos que $0 < N < 1$.

Además de las nuevas tramas, las estaciones también generan retransmisiones de tramas que con anterioridad sufrieron colisiones. Supongamos también que las tramas nuevas y antiguas combinadas están bien modeladas según una distribución de Poisson, con una media de G tramas por tiempo de trama. Es evidente que $G \geq N$. Con carga baja (es decir, $N \approx 0$) habrá pocas colisiones y, por lo tanto, pocas retransmisiones, por lo que $G \approx N$. Con carga alta habrá muchas colisiones, por lo que $G > N$. Con todas las cargas, la velocidad real de transmisión S es sólo la carga ofrecida, G, multiplicada por la probabilidad, P_0, de que una transmisión tenga éxito (es decir, $S = GP_0$, donde P_0 es la probabilidad de que una trama no sufra una colisión).

Una trama no sufrirá una colisión si no se envían otras tramas durante un tiempo de trama desde su envío, como se muestra en la figura 4-2. ¿Bajo qué condiciones llegará sin daño la trama sombreada? Sea t el tiempo requerido para enviar una trama. Si cualquier otro usuario generó una trama entre el tiempo t_0 y $t_0 + t$, el final de esa trama colisionará con el comienzo de la trama sombreada. De hecho, el destino de la trama sombreada está sentenciado aun antes de enviar el primer bit pero, dado que en ALOHA puro una estación no escucha el canal antes de transmitir, no tiene manera de saber que otra trama ya está en camino. Asimismo, cualquier otra trama que se inicie entre $t_0 + t$ y $t_0 + 2t$ chocará con el final de la trama sombreada.

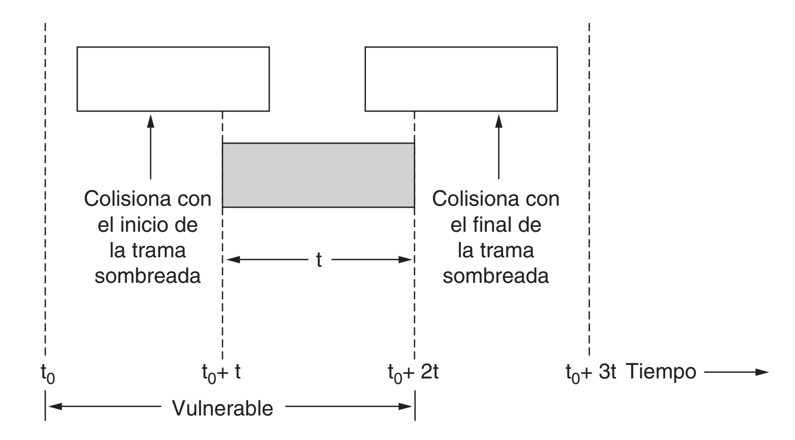


Figura 4-2. Periodo vulnerable para la trama sombreada.

La probabilidad de que se generen k tramas durante un tiempo de trama determinado, en donde se esperan G tramas, está dada por la distribución de Poisson:

$$\Pr[k] = \frac{G^k e^{-G}}{k!} \tag{4-2}$$

así, la probabilidad de cero tramas es simplemente e^{-G}. En un intervalo de dos tiempos de trama de longitud, el número promedio de tramas generadas es de $2G$. La probabilidad de que no se inicien tramas durante todo el periodo vulnerable está dada entonces por $P_0 = e^{-2G}$. Si $S = GP_0$, obtenemos:

$$S = Ge^{-2G}$$

En la figura 4-3 se muestra la relación entre el tráfico ofrecido y la velocidad real de transmisión. La máxima velocidad real de transmisión ocurre cuando $G = 0.5$, con $S = 1/2e$, que es alrededor de 0.184. En otras palabras, lo más que podemos esperar es un uso del canal de 18%. Este resultado no es muy alentador, pero con todo mundo transmitiendo al azar, difícilmente podríamos esperar una tasa de éxito de 100%.

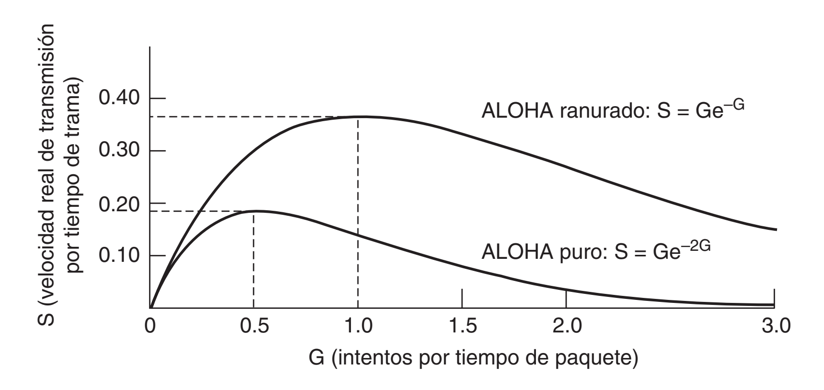


Figura 4-3. Velocidad real de transmisión contra tráfico ofrecido en los sistemas ALOHA.

ALOHA ranurado

Poco después de que ALOHA apareció en escena, Roberts (1972) publicó un método para duplicar la capacidad de un sistema ALOHA. Su propuesta fue dividir el tiempo en intervalos discretos llamados **ranuras**, cada uno de los cuales correspondía a una trama. Este método requiere que los usuarios acuerden límites de ranura. Una manera de lograr la sincronización sería tener una estación especial que emitiera una señal al comienzo de cada intervalo, como un reloj.

En el método de Roberts, que se conoce como **ALOHA ranurado**, en contraste con el **ALOHA puro** de Abramson, no se permite que una estación envíe cada vez que el usuario escribe una línea. En cambio, se le obliga a esperar el comienzo de la siguiente ranura. Por lo tanto, el ALOHA de tiempo continuo se convierte en uno de tiempo discreto. Esto reduce el periodo vulnerable a la mitad. Para ver esto, analice la figura 4-3 e imagine las colisiones que puede haber ahora. La probabilidad de que no haya más tráfico durante la misma ranura que nuestra trama de prueba es entonces de e^{-G}, lo que conduce a

$$S = Ge^{-G} \tag{4-3}$$

Como podemos ver en la figura 4-3, el ALOHA ranurado alcanza su máximo valor en $G = 1$, con una velocidad real de transmisión de $S = 1/e$, o aproximadamente 0.368, el doble que el ALOHA puro. Si el sistema está operando a $G = 1$, la probabilidad de una ranura vacía es de 0.368 (de la ecuación 4-2). Lo mejor que podemos esperar usando ALOHA ranurado es 37% de ranuras vacías, 37% de éxitos y 26% de colisiones. Si se opera con valores mayores de G se reduce el número de ranuras vacías pero aumenta de manera exponencial el número de colisiones. Para ver la manera en que se desarrolla este rápido crecimiento de colisiones con G, considere la transmisión de una trama de prueba. La probabilidad de que evitará una colisión es de e^{-G}, que es la probabilidad de que las demás estaciones estén en silencio durante esa ranura. La probabilidad de una colisión es entonces de sólo $1 - e^{-G}$. La probabilidad de que una transmisión requiera exactamente k intentos (es decir, $k - 1$ colisiones seguidas de un éxito) es

$$P_k = e^{-G}(1 - e^{-G})^{k-1}$$

El número esperado de transmisiones, E, por cada línea que se introduce en la terminal es

$$E = \sum_{k=1}^{\infty} kP_k = \sum_{k=1}^{\infty} ke^{-G}(1 - e^{-G})^{k-1} = e^{G}$$

Como resultado de la dependencia exponencial de E respecto a G, pequeños aumentos en la carga del canal pueden reducir drásticamente su desempeño.

El ALOHA ranurado es importante por una razón que al principio tal vez no sea obvia. Se diseñó en la década de 1970 y se utilizó en algunos sistemas experimentales iniciales, después casi se olvidó por completo. Cuando se inventó el acceso a Internet a través de cable, de repente surgió el problema de cómo asignar un canal compartido entre varios usuarios competidores. El ALOHA ranurado prácticamente se sacó del cesto de la basura para resolver el problema. Posteriormente, hacer que varias etiquetas RFID se comunicaran con el mismo lector RFID presentó otra variación del mismo problema. De nuevo salió al rescate el ALOHA ranurado, con unas cuantas ideas más mezcladas. Con frecuencia sucede que los protocolos que son perfectamente válidos caen en desuso por razones políticas (por ejemplo, alguna compañía grande desea que todos hagan las cosas a su manera) o debido a las tendencias siempre cambiantes de la tecnología. Después, años más tarde alguna persona astuta se dio cuenta de que un protocolo descartado por mucho tiempo es el que podía sacarlo de su problema actual. Por esta razón, en este capítulo estudiaremos varios protocolos elegantes que en la actualidad no se utilizan mucho, pero que podrían utilizarse fácilmente en aplicaciones futuras,

siempre y cuando los conozcan suficientes diseñadores de red. Por supuesto, también estudiaremos varios protocolos que se utilizan en la actualidad.

4.2.2 Protocolos de acceso múltiple con detección de portadora

Con el ALOHA ranurado, el mejor aprovechamiento de canal que se puede lograr es $1/e$. Este resultado tan bajo no es muy sorprendente pues, con estaciones que transmiten a voluntad propia, sin prestar atención a lo que están haciendo las demás estaciones, es inevitable que haya muchas colisiones. Sin embargo, en las redes LAN es posible que las estaciones detecten lo que están haciendo las demás estaciones y adapten su comportamiento con base en ello. Estas redes pueden lograr una utilización mucho mejor que $1/e$. En esta sección estudiaremos algunos protocolos para mejorar su desempeño.

Los protocolos en los que las estaciones escuchan una portadora (es decir, una transmisión) y actúan de manera acorde se llaman **protocolos de detección de portadora**. Se han propuesto varios de ellos y hace mucho tiempo se analizaron con detalle. Por ejemplo, consulte a Kleinrock y Tobagi (1975). A continuación mencionaremos varias versiones de los protocolos de detección de portadora.

CSMA persistente y no persistente

El primer protocolo de detección de portadora que estudiaremos aquí se llama **CSMA** (**Acceso Múltiple con Detección de Portadora**, del inglés *Carrier Sense Multiple Access*) **persistente-1**. Es un nombre bastante largo para el esquema CSMA más simple. Cuando una estación tiene datos por enviar, primero escucha el canal para saber si alguien más está transmitiendo en ese momento. Si el canal está inactivo, la estación envía sus datos. Por el contrario, si el canal está ocupado, la estación espera hasta que se desocupa. A continuación, la estación transmite una trama. Si ocurre una colisión, la estación espera una cantidad aleatoria de tiempo y comienza de nuevo. El protocolo se llama persistente-1 porque la estación transmite con una probabilidad de 1 cuando encuentra que el canal está inactivo.

Podría esperarse que este esquema evite las colisiones, excepto en el extraño caso de los envíos simultáneos, pero de hecho no lo hace. Si dos estaciones están listas a la mitad de la transmisión de una tercera estación, ambas esperarán amablemente hasta que termine la transmisión y después ambas empezarán a transmitir exactamente al mismo tiempo, lo cual producirá una colisión. Si no fueran tan impacientes, habría menos colisiones.

Otro aspecto delicado es que el retardo de propagación tiene un efecto importante sobre las colisiones. Existe la posibilidad de que, justo después de que una estación comienza a transmitir, otra estación esté lista para enviar y detecte el canal. Si la señal de la primera estación no ha llegado aún a la segunda, esta última detectará un canal inactivo y comenzará también a enviar, lo que dará como resultado una colisión. Esta posibilidad depende del número de tramas que quepan en el canal, o **producto de ancho de banda-retardo** del canal. Si sólo cabe una pequeña fracción de una trama en el canal, lo cual es cierto en la mayoría de las redes LAN, ya que el retardo de propagación es pequeño, la posibilidad de que ocurra una colisión es pequeña. Cuanto mayor sea el producto de ancho de banda-retardo, más importante será este efecto y peor el desempeño del protocolo.

Aun así, este protocolo tiene un mejor desempeño que el ALOHA puro, ya que ambas estaciones tienen la decencia de dejar de interferir con la trama de la tercera estación. Lo mismo se aplica en el ALOHA ranurado.

Un segundo protocolo de detección de portadora es el **CSMA no persistente**. En este protocolo se hace un intento consciente por ser menos egoísta que en el previo. Como antes, una estación escucha el canal cuando desea enviar una trama y, si nadie más está transmitiendo, comienza a hacerlo. Pero si el canal ya está en uso, la estación no lo escuchará de manera continua con el fin de tomarlo de inmediato al

detectar el final de la transmisión anterior, sino que esperará un periodo aleatorio y repetirá el algoritmo. En consecuencia, este algoritmo conduce a un mejor uso del canal pero produce mayores retardos que el CSMA persistente-1.

El último protocolo es el **CSMA persistente-p**, que se aplica a canales ranurados y funciona como se explica a continuación. Cuando una estación está lista para enviar, escucha el canal. Si se encuentra inactivo, la estación transmite con una probabilidad p. Con una probabilidad $q = 1 - p$, se posterga hasta la siguiente ranura. Si esa ranura también está inactiva, la estación transmite o posterga una vez más, con probabilidades p y q. Este proceso se repite hasta que se transmite la trama o hasta que otra estación comienza a transmitir. En el segundo caso, la desafortunada estación actúa como si hubiera ocurrido una colisión (es decir, espera un tiempo aleatorio y comienza de nuevo). Si al principio la estación detecta que el canal está ocupado, espera hasta la siguiente ranura y aplica el algoritmo anterior. El estándar IEEE 802.11 usa una versión refinada del CSMA persistente-p que veremos en la sección 4.4.

En la figura 4-4 se muestra la velocidad real de transmisión calculada contra el tráfico ofrecido para los tres protocolos, así como para el ALOHA puro y el ranurado.

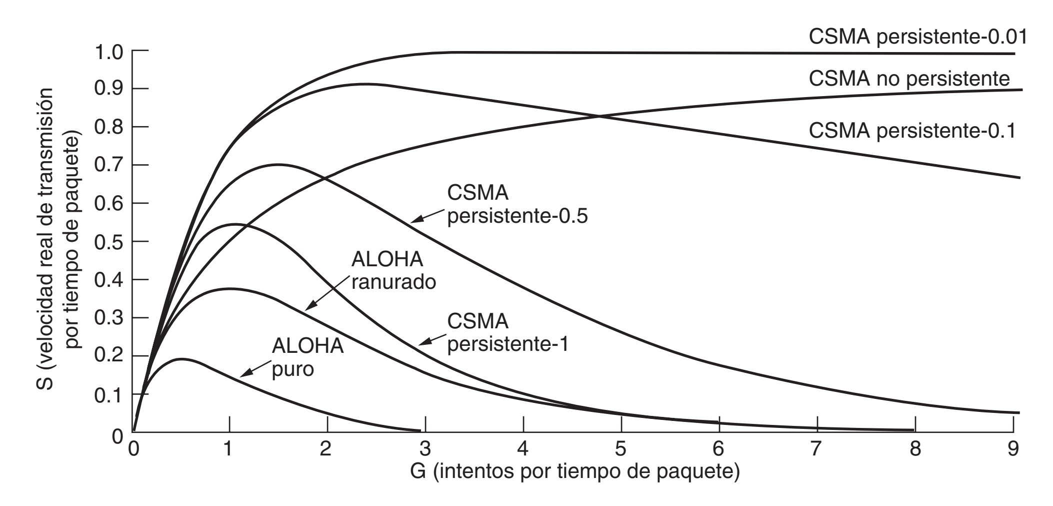


Figura 4-4. Comparación de la utilización del canal contra la carga para varios protocolos de acceso aleatorio.

CSMA con detección de colisiones

En definitiva, los protocolos CSMA persistentes y no persistentes son una mejora respecto a ALOHA porque aseguran que ninguna estación empezará a transmitir mientras el canal esté ocupado. Pero si dos estaciones detectan que el canal está inactivo y empiezan a transmitir al mismo tiempo, sus señales de todas formas sufrirán una colisión. Otra mejora es que las estaciones detecten rápidamente la colisión y dejen de transmitir de inmediato (en vez de terminadas las transmisiones), ya que de todas formas se alterarán y no se podrán recuperar. Esta estrategia ahorra tiempo y ancho de banda.

Este protocolo, conocido como **CSMA/CD** (**CSMA con Detección de Colisiones**, del inglés *CSMA with Collision Detection*), es la base de la clásica LAN Ethernet, por lo que vale la pena dedicar un poco de tiempo a analizarlo con detalle. Es importante tener en cuenta que la detección de colisiones es un proceso analógico. El hardware de la estación debe escuchar el canal mientras transmite. Si la señal que recibe es distinta de la señal que está enviando, sabe que está ocurriendo una colisión. Las implicaciones son que una señal recibida no debe ser pequeña en comparación con la señal transmitida (lo cual es difícil en las redes inalámbricas, ya que las señales recibidas pueden ser 1 000 000 de veces más débiles que las

señales transmitidas) y que la modulación se debe elegir de modo que permita detectar colisiones (por ejemplo, tal vez sea imposible detectar una colisión de dos señales de 0 volts).

Al igual que muchos otros protocolos de LAN, CSMA/CD utiliza el modelo conceptual de la figura 4-5. En el punto marcado como t_0, una estación ha terminado de transmitir su trama. Cualquier otra estación que tenga una trama por enviar puede intentar hacerlo ahora. Si dos o más estaciones deciden transmitir en forma simultánea, habrá una colisión. Si una estación detecta una colisión, aborta la transmisión, espera un tiempo aleatorio e intenta de nuevo (suponiendo que ninguna otra estación ha comenzado a transmitir durante ese lapso). Por lo tanto, nuestro modelo de CSMA/CD consistirá en periodos alternantes de contención y transmisión, con periodos de inactividad que ocurrirán cuando todas las estaciones estén en reposo (por ejemplo, por falta de trabajo).

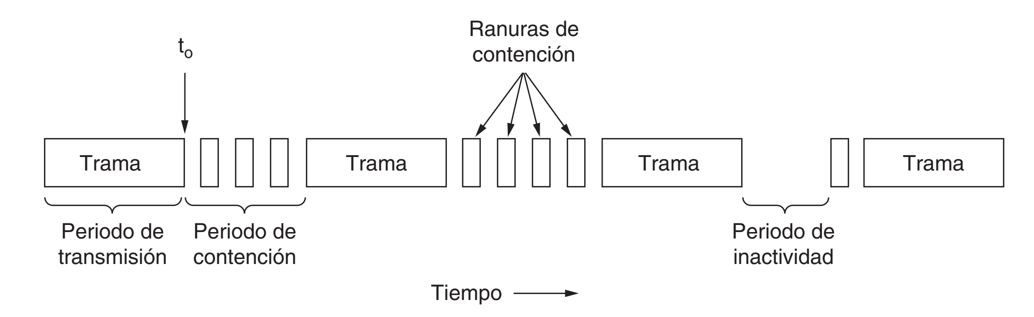


Figura 4-5. CSMA/CD puede estar en estado de contención, de transmisión o inactivo.

Ahora observemos con cuidado los detalles del algoritmo de contención. Suponga que dos estaciones comienzan a transmitir exactamente en el momento t_0. ¿En cuánto tiempo se darán cuenta de que ha ocurrido una colisión? La respuesta a esta pregunta es vital para determinar la longitud del periodo de contención y, por lo tanto, el retardo y la velocidad real de transmisión.

El tiempo mínimo para detectar la colisión es tan sólo el tiempo que tarda la señal en propagarse de una estación a otra. Con base en esta información, podríamos pensar que una estación que no ha detectado una colisión durante un periodo igual al tiempo completo de propagación del cable después de iniciar su transmisión puede estar segura de que ha tomado el cable. Por "tomado" queremos decir que todas las demás estaciones saben que está transmitiendo y no interferirán. Esta conclusión es errónea.

Considere el siguiente escenario en el peor caso. Sea τ el tiempo que tarda una señal en propagarse entre las dos estaciones más lejanas. En t_0, una estación comienza a transmitir. En $t_0 + \tau - \varepsilon$, un instante antes de que la señal llegue a la estación más lejana, esa estación también comienza a transmitir. Por supuesto que detecta la colisión casi de inmediato y se detiene, pero la pequeña ráfaga de ruido causada por la colisión no regresa a la estación original, sino hasta el tiempo $2\tau - \varepsilon$. En otras palabras, en el peor caso una estación no puede estar segura de que ha tomado el canal hasta que ha transmitido durante 2τ sin detectar una colisión.

Con este razonamiento, podemos pensar en la contención de CSMA/CD como un sistema ALOHA ranurado con un ancho de ranura de 2τ. En un cable coaxial de 1 km de longitud, $\tau \approx 5$ μseg. La diferencia para CSMA/CD en comparación con ALOHA ranurado es que las ranuras en las que sólo transmite una estación (por ejemplo, la que tomó el canal) van seguidas del resto de una trama. Esta diferencia mejorará en forma considerable el desempeño si el tiempo de la trama es mucho mayor que el tiempo de propagación.

4.2.3 Protocolos libres de colisiones

Aunque las colisiones no ocurren en CSMA/CD una vez que una estación ha capturado el canal sin ambigüedades, aún pueden ocurrir durante el periodo de contención. Estas colisiones afectan en forma adversa el desempeño del sistema, en especial cuando el producto ancho de banda-retardo es grande, como cuando el cable es largo (es decir, τ es grande) y las tramas son cortas. Las colisiones no sólo reducen el ancho de banda, sino que también hacen variable el tiempo de envió de una trama, lo cual no es bueno para el tráfico en tiempo real tal como la voz sobre IP. Además, CSMA/CD no se puede aplicar en forma universal.

En esta sección examinaremos algunos protocolos que resuelven la contención por el canal sin que haya colisiones, ni siquiera durante el periodo de contención. En la actualidad, la mayoría de estos protocolos no se utilizan en los sistemas grandes, pero en un campo en constante cambio, el hecho de tener algunos protocolos con excelentes propiedades disponibles para sistemas futuros es con frecuencia algo bueno.

En los protocolos que describiremos supondremos que hay exactamente N estaciones, cada una programada con una dirección única de 0 a $N - 1$. No importa el hecho de que algunas estaciones puedan estar inactivas una parte del tiempo. También damos por hecho que el retardo de propagación es insignificante. La pregunta básica persiste: ¿qué estación obtiene el canal después de una transmisión exitosa? Seguimos usando el modelo de la figura 4-5 con sus ranuras de contención discretas.

Un protocolo de mapa de bits

En nuestro primer protocolo libre de colisiones, el **método básico de mapa de bits**, cada periodo de contención consiste exactamente de N ranuras. Si la estación 0 tiene una trama por enviar, transmite un bit 1 durante la ranura 0. No está permitido a ninguna otra estación transmitir durante esta ranura. Sin importar lo que haga la estación 0, la estación 1 tiene la oportunidad de transmitir un bit 1 durante la ranura 1, pero sólo si tiene una trama puesta en la cola. En general, la estación j puede anunciar que tiene una trama por enviar, para lo cual inserta un bit 1 en la ranura j. Una vez que han pasado las N ranuras, cada estación tiene un completo conocimiento acerca de cuáles son las estaciones que quieren transmitir. En ese punto, las estaciones empiezan a transmitir en orden numérico (vea la figura 4-6).

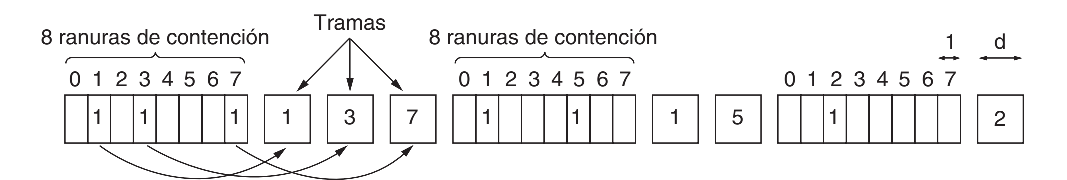


Figura 4-6. El protocolo básico de mapa de bits.

Como todos están de acuerdo en quién sigue a continuación, nunca habrá colisiones. Una vez que la última estación lista haya transmitido su trama, un evento que pueden detectar fácilmente todas las estaciones, comienza otro periodo de contención de N bits. Si una estación está lista justo después de que ha pasado su ranura de bit, ha tenido mala suerte y deberá permanecer inactiva hasta que cada una de las demás estaciones haya tenido su oportunidad y el mapa de bits haya comenzado de nuevo.

Los protocolos como éste en los que el interes de transmitir se difunde antes de la transmisión se llaman **protocolos de reservación**, debido a que reservan la propiedad del canal por anticipado y evitan

colisiones. Analicemos brevemente el desempeño de este protocolo. Por conveniencia, mediremos el tiempo en unidades de la ranura de bit de contención, con tramas de datos consistentes en d unidades de tiempo.

En condiciones de carga baja, el mapa de bits simplemente se repetirá una y otra vez, debido a la falta de tramas de datos. Considere la situación desde el punto de vista de una estación de menor numeración, como 0 o 1. Por lo general, cuando la estación está lista para transmitir, la ranura "actual" estará en algún lugar a la mitad del mapa de bits. En promedio, la estación tendrá que esperar $N/2$ ranuras para que el escaneo actual termine, además de otras N ranuras para que el siguiente escaneo se ejecute hasta su terminación, antes de que pueda empezar a transmitir.

Las posibilidades para las estaciones de mayor numeración son mejores. En general, éstas sólo tendrán que esperar la mitad de un escaneo ($N/2$ ranuras de bits) antes de comenzar a transmitir. Las estaciones de mayor numeración pocas veces tienen que esperar el siguiente escaneo. Dado que las estaciones de menor numeración deben esperar en promedio $1.5N$ ranuras y las estaciones de mayor numeración esperar en promedio $0.5N$ ranuras, la media de todas las estaciones es de N ranuras.

La eficiencia del canal cuando la carga es baja es fácil de calcular. La sobrecarga por trama es de N bits y la cantidad de datos es de d bits, lo cual nos da una eficiencia de $d/(d + N)$.

Si la carga es alta y todas las estaciones tienen algo que enviar todo el tiempo, el periodo de contención de N bits se prorratea entre N tramas, lo cual produce una sobrecarga de sólo 1 bit por trama, o una eficiencia de $d/(d + 1)$. El retardo promedio de una trama es igual a la suma del tiempo que está en cola en su estación, más un $(N - 1)d + N$ adicional una vez que llega a la cabeza de su cola interna. Este intervalo indica cuánto tiempo hay que esperar a que las demás estaciones tomen su turno para enviar una trama y otro mapa de bits.

Paso de token

La esencia del protocolo de mapa de bits es que permite que cada estación transmita una trama por turno, en un orden predefinido. Otra forma de lograr lo mismo es pasar un pequeño mensaje conocido como **token** de una estación a otra, en el mismo orden predefinido. El token representa el permiso para enviar. Si una estación tiene una trama puesta en cola para transmitirla cuando recibe el token, puede enviar esa trama antes de pasar el token a la siguiente estación. Si no tiene una trama puesta en cola, simplemente pasa el token.

En un protocolo **token ring**, la topología de la red se utiliza para definir el orden en el que las estaciones envían información. Las estaciones están conectadas una con otra en un solo anillo. Así, el proceso de pasar el token a la siguiente estación consiste en recibir el token proveniente de una dirección y transmitirlo hacia la otra dirección, como podemos ver en la figura 4-7. Las tramas también se transmiten en la dirección del token. De esta forma, circularán alrededor del anillo y llegarán a la estación de destino. Sin embargo, para evitar que la trama circule en forma indefinida (como el token), una estación necesita quitarla del anillo. Esta estación puede ser la que envió originalmente la trama, después de que haya pasado por un ciclo completo, o la estación destinada a recibir la trama.

Cabe mencionar que no necesitamos un anillo físico para implementar el paso del token. El canal que conecta a las estaciones podría ser también un solo bus extenso. Así, cada estación puede usar el bus para enviar el token a la siguiente estación en la secuencia predefinida. Al poseer el token, una estación puede usar el bus para enviar una trama, como antes. A este protocolo se le conoce como **token bus**.

El desempeño del protocolo de paso de token es similar al del protocolo de mapa de bits, aunque las ranuras de contención y las tramas de un ciclo están ahora entremezcladas. Después de enviar una trama, cada estación debe esperar a que las N estaciones (incluyéndose a sí misma) envíen el token a sus estaciones vecinas y que las otras $N - 1$ estaciones envíen una trama, si es que la tienen. Una sutil diferencia es que, como todas las posiciones en el ciclo son equivalentes, no hay parcialidad por las estaciones de

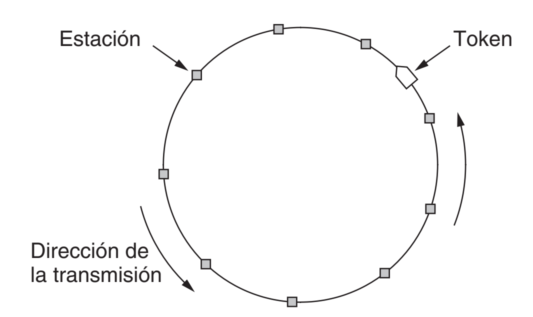


Figura 4-7. El protocolo *token ring*.

menor o de mayor numeración. Para token ring, cada estación también envía el token sólo hasta su estación vecina antes de que el protocolo lleve a cabo el siguiente paso. No es necesario propagar cada token a todas las estaciones antes de que el protocolo avance al siguiente paso.

Las redes de token ring han surgido como protocolos MAC con cierta consistencia. Uno de los primeros protocolos de este tipo (conocido como "Token Ring", que se estandarizó como IEEE 802.5) fue popular en la década de 1980 como alternativa a la Ethernet clásica. En la década de 1990, una red token ring mucho más veloz conocida como **FDDI** (**Interfaz de Datos Distribuidos por Fibra**, del inglés *Fiber Distributed Data Interface*) fue vencida por la Ethernet conmutada. En la década de 2000, una red token ring llamada **RPR** (**Anillo de Paquetes con Recuperación**, del inglés *Resilent Packet Ring*) se definió como el IEEE 802.17 para estandarizar la mezcla de anillos de área metropolitana que usaban los ISP. Nos preguntamos: ¿Qué nos ofrecerá la década de 2010?.

Conteo descendente binario

Un problema con el protocolo básico de mapa de bits, y en consecuencia con el paso de token, es que la sobrecarga es de 1 bit por estación, por lo que no se escala bien en redes con miles de estaciones. Podemos tener mejores resultados si usamos direcciones de estación binarias con un canal que combine las transmisiones. Una estación que quiere utilizar el canal en un momento dado difunde su dirección como una cadena binaria de bits, comenzando por el bit de mayor orden. Se supone que todas las direcciones tienen la misma longitud. A todos los bits en cada posición de dirección de las diferentes estaciones se les aplica un OR BOOLEANO por el canal cuando se envían al mismo tiempo. A este protocolo lo llamaremos **conteo descendente binario**. Se utilizó en Datakit (Fraser, 1987). Asume de manera implícita que los retardos de transmisión son insignificantes, de manera que todas las estaciones ven los bits instantáneamente.

Para evitar conflictos, es necesario aplicar una regla de arbitraje: tan pronto como una estación ve que una posición de bit de orden alto, cuya dirección es 0, ha sido sobrescrita con un 1, se da por vencida. Por ejemplo, si las estaciones 0010, 0100, 1001 y 1010 están tratando de obtener el canal, en el primer tiempo de bit las estaciones transmiten 0, 0, 1 y 1, respectivamente. A éstos se les aplica el OR para formar un 1. Las estaciones 0010 y 0100 ven el 1 y saben que una estación de mayor numeración está compitiendo por el canal, por lo que se dan por vencidas durante esta ronda. Las estaciones 1001 y 1010 continúan.

El siguiente bit es 0, y ambas estaciones continúan. El siguiente bit es 1, por lo que la estación 1001 se da por vencida. La ganadora es la estación 1010, debido a que tiene la dirección más alta. Después de ganar la contienda, ahora puede transmitir una trama, después de lo cual comienza otro ciclo de contienda. El protocolo se ilustra en la figura 4-8. Tiene la propiedad de que estaciones con mayor numeración tienen una prioridad más alta que las estaciones con menor numeración, lo cual puede ser bueno o malo, dependiendo del contexto.

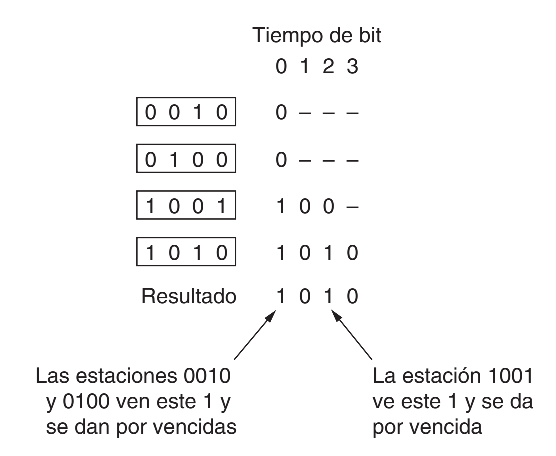


Figura 4-8. El protocolo de conteo descendente binario. Un guión indica un silencio.

La eficiencia de canal de este método es de $d/(d + \log_2 N)$. Pero si el formato de trama se escoge ingeniosamente de modo que la dirección del emisor sea el primer campo en la trama, ni siquiera estos $\log_2 N$ bits se desperdician y la eficiencia es del 100%.

El conteo descendente binario es un ejemplo de un protocolo sencillo, elegante y eficiente que está esperando a ser redescubierto. Esperemos que algún día encuentre un nuevo hogar.

4.2.4 Protocolos de contención limitada

Hasta ahora hemos considerado dos estrategias básicas para adquirir el canal en una red de difusión: los protocolos de contención, como el CSMA, y los protocolos libres de colisión. Cada estrategia se puede recomendar según lo bien que funciona en relación con las dos medidas importantes de desempeño: el retardo con carga baja y la eficiencia del canal con carga alta. En condiciones de carga ligera, la contención (es decir, ALOHA puro o ranurado) es preferible debido a su bajo retardo (ya que las colisiones son raras). A medida que aumenta la carga, la contención se vuelve cada vez menos atractiva, debido a que la sobrecarga asociada al arbitraje del canal se vuelve mayor. Lo inverso se cumple para los protocolos libres de colisiones. Con carga baja tienen un retardo alto, pero a medida que aumenta la carga mejora la eficiencia del canal (ya que las sobrecargas son fijas).

Sin duda, sería agradable si pudiéramos combinar las mejores propiedades de los protocolos de contención y los libres de colisiones, para idear un nuevo protocolo que usara contención cuando la carga fuera baja y con ello tener un retardo bajo, así como una técnica libre de colisiones cuando la carga fuera alta para lograr una buena eficiencia de canal. De hecho existen tales protocolos, a los que llamaremos **protocolos de contención limitada**, y son con los que concluiremos nuestro estudio de las redes de detección de portadora.

Hasta ahora, los únicos protocolos de contención que hemos estudiado han sido simétricos. Es decir, cada estación intenta adquirir el canal con cierta probabilidad, p, y todas las estaciones usan la misma p. Resulta interesante que el desempeño general del sistema se pueda mejorar a veces mediante el uso de un protocolo que asigne diferentes probabilidades a distintas estaciones.

Antes de analizar los protocolos asimétricos, demos un breve repaso al desempeño del caso simétrico. Suponga que hay k estaciones que compiten por el acceso al canal. Cada estación tiene una probabilidad p de transmitir durante cada ranura. La probabilidad de que una estación adquiera con éxito el canal durante una ranura dada es la probabilidad de que cualquier otra estación transmita, con probabilidad p, y que las

otras $k - 1$ estaciones posterguen su transmisión, cada una con una probabilidad de $1 - p$. Este valor es $kp(1 - p)^{k-1}$. Para encontrar el valor óptimo de p, diferenciamos con respecto a p, igualamos el resultado a cero y despejamos p. Al hacer esto, encontramos que el mejor valor de p es $1/k$. Si sustituimos $p = 1/k$, obtenemos:

$$\Pr[\text{éxitos con } p \text{ óptima}] = \left[\frac{k-1}{k}\right]^{k-1} \tag{4-4}$$

En la figura 4-9 se grafica esta probabilidad. Para un número pequeño de estaciones, la posibilidad de éxito es buena, pero tan pronto como la cantidad de estaciones llega a cinco, la probabilidad disminuye hasta una cifra cercana a su valor asintótico de $1/e$.

En la figura 4-9 es bastante evidente que la probabilidad de que una estación adquiera el canal sólo puede aumentar si disminuye la cantidad de competencia. Los protocolos de contención limitada hacen precisamente eso. Primero dividen las estaciones en grupos (no necesariamente separados). Sólo los miembros del grupo 0 pueden competir por la ranura 0. Si uno de ellos tiene éxito, adquiere el canal y transmite su trama. Si la ranura permanece desocupada o si hay una colisión, los miembros del grupo 1 compiten por la ranura 1, etcétera. Al dividir en forma adecuada las estaciones en grupos, es posible reducir la cantidad de contenciones para cada ranura y, en consecuencia, se puede operar cada ranura cerca de la parte izquierda de la figura 4-9.

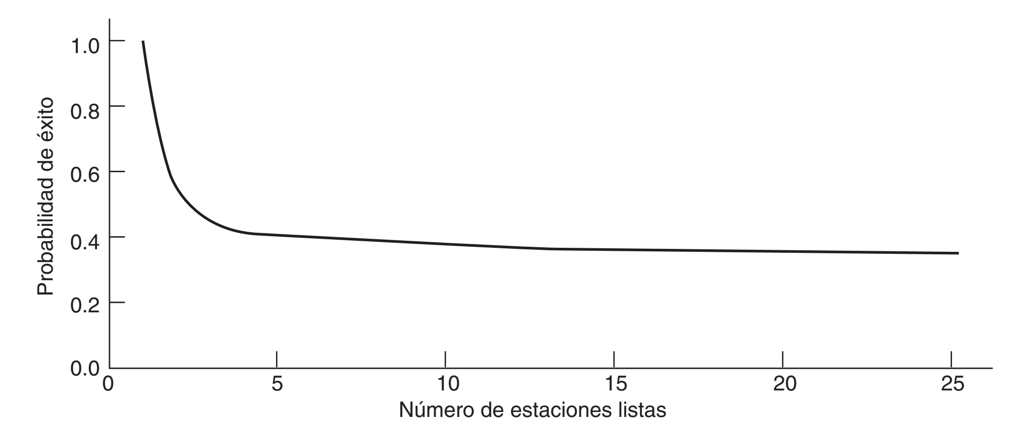


Figura 4-9. Probabilidad de adquisición de un canal de contención simétrica.

El truco está en cómo asignar las estaciones a las ranuras. Antes de ver el caso general, consideremos algunos casos especiales. En un extremo, cada grupo tiene sólo un miembro. Una asignación de este tipo garantiza que nunca habrá colisiones, pues a lo más sólo una estación competirá por una ranura dada. Ya hemos visto tales protocolos (por ejemplo, el conteo descendente binario). El siguiente caso especial es asignar dos estaciones por grupo. La probabilidad de que ambas intenten transmitir durante una ranura es p^2, que para una p pequeña es insignificante. A medida que se asignan cada vez más estaciones a la misma ranura, aumenta la probabilidad de colisión pero disminuye la longitud del escaneo del mapa de bits necesaria para dar a todos una oportunidad. El caso límite es un solo grupo que contenga todas las estaciones (es decir, ALOHA ranurado). Lo que necesitamos es una manera de asignar dinámicamente las estaciones a las ranuras, con muchas estaciones por ranura cuando la carga es baja y pocas estaciones (o incluso sólo una) por ranura cuando la carga es alta.

El protocolo de recorrido de árbol adaptable

Una manera muy sencilla de llevar a cabo la asignación necesaria es usar el algoritmo desarrollado por el ejército de Estados Unidos para hacer pruebas de sífilis a los soldados durante la Segunda Guerra Mundial (Dorfman, 1943). En esencia, el ejército tomaba una muestra de sangre de *N* soldados. Se vaciaba una parte de cada muestra en un solo tubo de ensayo. Luego se examinaba esta muestra mezclada en busca de anticuerpos. Si no se encontraban, todos los soldados del grupo se declaraban sanos. Si se encontraban anticuerpos, se preparaban dos nuevas muestras mezcladas, una de los soldados 1 a *N*/2 y otra de los demás. El proceso se repetía en forma recursiva hasta que se determinaban los soldados infectados.

Para la versión de computadora de este algoritmo (Capetanakis, 1979) es conveniente considerar a las estaciones como hojas de un árbol binario, como se muestra en la figura 4-10. En la primera ranura de contención después de la transmisión exitosa de una trama (ranura 0), se permite que todas las estaciones intenten adquirir el canal. Si una de ellas lo logra, qué bueno. Si hay una colisión, entonces durante la ranura 1, sólo aquellas estaciones que queden bajo el nodo 2 del árbol podrán competir. Si alguna de ellas adquiere el canal, la ranura que siga después de la trama se reservará para las estaciones que están bajo el nodo 3. Por otra parte, si dos o más estaciones bajo el nodo 2 quieren transmitir, habrá una colisión durante la ranura 1, en cuyo caso será el turno del nodo 4 durante la ranura 2.

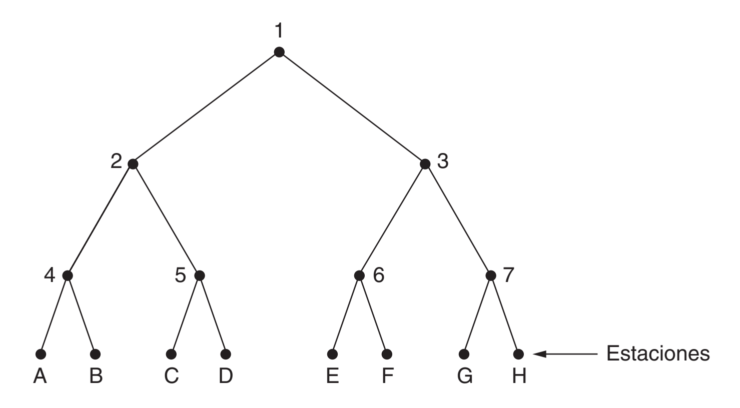


Figura 4-10. El árbol para ocho estaciones.

En esencia, si ocurre una colisión durante la ranura 0, se examina todo el árbol para localizar todas las estaciones listas. Cada ranura de bits está asociada a un nodo específico del árbol. Si ocurre una colisión, continúa la búsqueda en forma recursiva con el hijo izquierdo y el derecho del nodo. Si una ranura de bits está inactiva o si sólo una estación que transmite en ella, se puede detener la búsqueda de su nodo, ya que se han localizado todas las estaciones listas (si hubiera existido más de una, habría ocurrido una colisión).

Cuando la carga del sistema es pesada, apenas si vale la pena dedicarle la ranura 0 al nodo 1, porque eso sólo tiene sentido en el caso poco probable de que haya precisamente una estación que tenga una trama por enviar. Asimismo, podríamos argumentar que sería conveniente saltar los nodos 2 y 3 por la misma razón. En términos más generales, ¿en qué nivel del árbol debe comenzar la búsqueda? Es obvio que a mayor carga, la búsqueda debe comenzar más abajo en el árbol. Supondremos que cada estación tiene una buena estimación del número de estaciones listas, *q*, por ejemplo, mediante una supervisión del tráfico reciente.

Para proceder, vamos a numerar los niveles del árbol desde arriba, con el nodo 1 de la figura 4-10 en el nivel 0, los nodos 2 y 3 en el nivel 1, etcétera. Observe que cada nodo del nivel *i* tiene una fracción 2^{-i}

de las estaciones por debajo de él. Si las q estaciones listas se distribuyen de manera uniforme, el número esperado de ellas por debajo de un nodo específico en el nivel i es de sólo $2^{-i}q$. Instintivamente esperaríamos que el nivel óptimo para comenzar a examinar al árbol fuera aquel cuyo número promedio de estaciones contendientes por ranura sea 1; es decir, el nivel en el que $2^{-i}q = 1$. Al resolver esta ecuación, encontramos que $i = \log_2 q$.

Se han descubierto muchas mejoras al algoritmo básico, las cuales se han analizado con cierto detalle por Bertsekas y Gallager (1992). Por ejemplo, considere el caso en el que las estaciones G y H son las únicas que quieren transmitir. En el nodo 1 ocurrirá una colisión, por lo que se intentará el 2, pero se encontrará inactivo. No tiene caso probar el nodo 3, ya que está garantizado que tendrá una colisión (sabemos que dos o más estaciones bajo 1 están listas y que ninguna de ellas está bajo 2, por lo que todas deben estar bajo 3). Podemos omitir la prueba de 3 para intentar con el nodo 6. Al no arrojar nada esta prueba, también podemos omitir el nodo 7 para intentar el nodo G después.

4.2.5 Protocolos de LAN inalámbrica

Un sistema de computadoras portátiles que se comunican por radio se puede considerar una LAN inalámbrica, como vimos en la sección 1.5.3. Este tipo de LAN es un ejemplo de un canal de difusión. Además, tiene propiedades un tanto diferentes que la LAN alámbrica, por lo que requiere distintos protocolos MAC. En esta sección, examinaremos algunos de estos protocolos. En la sección 4.4 analizaremos el 802.11 (WiFi) con detalle.

Una configuración común para una LAN inalámbrica es un edificio de oficinas con puntos de acceso (AP) ubicados de manera estratégica alrededor del edificio. Los AP están interconectados mediante cobre o fibra, y proveen conectividad a las estaciones que se comunican con ellos. Si la potencia de transmisión de los AP y las computadoras portátiles se ajusta de modo que tenga un alcance de decenas de metros, entonces los cuartos cercanos se convierten en una celda única y el edificio entero se convierte en algo así como el sistema de telefonía celular que estudiamos en el capítulo 2, excepto que cada celda sólo tiene un canal. Todas las estaciones en la celda comparten este canal, incluyendo el AP. Por lo general proporciona anchos de banda de varios megabits/seg, hasta 600 Mbps.

Ya hemos recalcado de antemano que los sistemas inalámbricos no pueden por lo general detectar una colisión al momento en que ocurre. La señal recibida en una estación puede ser débil, tal vez un millón de veces más tenue que la señal transmitida. Encontrarla es como buscar una aguja en un pajar. En vez de ello se utilizan confirmaciones de recepción para descubrir las colisiones y otros errores después de que suceden.

Incluso hay una diferencia aún más importante entre las redes LAN inalámbricas y las LAN alámbricas (cableadas). Tal vez una estación en una LAN inalámbrica no pueda transmitir ni recibir tramas de todas las demás estaciones debido al rango de radio limitado de éstas. En las redes LAN alámbricas, cuando una estación envía una trama, todas las demás estaciones la reciben. La ausencia de esta propiedad en las redes LAN inalámbricas provoca una variedad de complicaciones.

En nuestros siguientes análisis haremos la suposición de simplificación de que cada transmisor de radio tiene cierto rango físico, el cual se representa mediante una región de cobertura circular dentro de la cual otra estación puede detectar y recibir la transmisión de la estación emisora. Es importante darse cuenta de que en la práctica, las regiones casi nunca son tan regulares debido a que la propagación de las señales de radio depende del entorno. Las paredes y otros obstáculos que atenúan y reflejan señales pueden hacer que el alcance difiera de manera considerable en distintas direcciones. Pero un simple modelo circular bastará para nuestros propósitos.

Un enfoque inocente para usar una LAN inalámbrica podría ser probar con CSMA: escuchar si hay otras transmisiones y sólo transmitir si nadie más lo está haciendo. El problema radica en que este proto-

colo no es en realidad una buena manera de pensar en lo inalámbrico, ya que lo que importa en la recepción es la interferencia en el receptor, no en el emisor. Para ver la naturaleza de este problema, considere la figura 4-11, en la que se ilustran cuatro estaciones inalámbricas. Para nuestros fines, no importa cuáles son AP ni cuáles son computadoras portátiles. El alcance de radio es tal que *A* y *B* están en el mismo alcance y es probable que puedan interferir entre sí. *C* también podría interferir con *B* y con *D*, pero no con *A*.

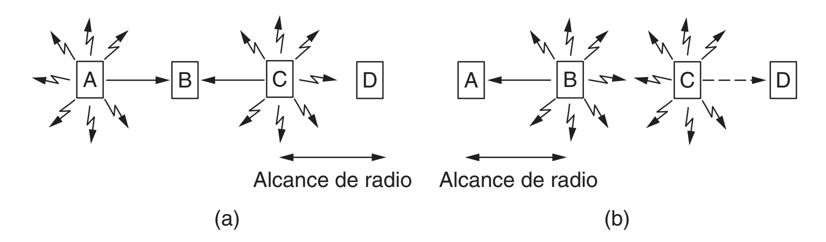


Figura 4-11. Una LAN inalámbrica. (a) *A* y *C* son terminales ocultas cuando transmiten a *B*. (b) *B* y *C* son terminales expuestas cuando transmiten a *A* y *D*.

Primero considere lo que ocurre cuando *A* y *C* transmiten hacia *B*, como se muestra en la figura 4-11(a). Si *A* envía y *C* detecta el medio de inmediato, no podrá escuchar a *A* porque está fuera de su alcance. Por lo tanto, *C* concluirá falsamente que puede transmitir a *B*. Si *C* comienza a transmitir, interferirá en *B*, eliminando la trama de *A* (en este caso suponemos que no se utiliza un esquema tipo CDMA para proveer múltiples canales, por lo que las colisiones alteran la señal y destruyen ambas tramas). Queremos un protocolo MAC que evite que ocurra este tipo de colisión, debido a que desperdicia ancho de banda. El problema de que una estación no pueda detectar a un competidor potencial por el medio, debido a que dicho competidor está demasiado lejos, se denomina **problema de terminal oculta**.

Ahora consideremos una situación distinta: *B* transmite a *A* al mismo tiempo que *C* desea transmitir a *D*, como se muestra en la figura 4-11(b). Si *C* detecta el medio, escuchará una transmisión y concluirá equivocadamente que no puede enviar a *D* (lo cual se muestra con una línea punteada). De hecho, esa transmisión provocaría una mala recepción sólo en la zona entre *B* y *C*, en donde no hay ninguno de los receptores deseados. Queremos un protocolo MAC que evite que ocurra este tipo de aplazamiento, ya que desperdicia ancho de banda. A esta situación se le denomina **problema de terminal expuesta**.

El problema es que antes de comenzar una transmisión, una estación realmente quiere saber si hay actividad o no alrededor del receptor. El CSMA simplemente le indica si hay o no actividad cerca del transmisor mediante la detección de la portadora. Con un cable, todas las señales se propagan a todas las estaciones, por lo que esta distinción no existe. Sin embargo, sólo puede llevarse a cabo una transmisión en un momento dado en cualquier lugar del sistema. En un sistema basado en ondas de radio de corto alcance, pueden ocurrir transmisiones simultáneas si las ondas tienen destinos diferentes y éstos están fuera de alcance entre sí. Queremos que esta concurrencia ocurra a medida que aumenta el tamaño de la celda, de la misma forma que las personas en una fiesta no deben esperar a que todos en el cuarto hagan silencio para poder hablar; puede haber múltiples conversaciones a la vez en un cuarto grande, siempre y cuando no estén dirigidas hacia la misma ubicación.

Uno de los primeros protocolos influyentes que aborda estos problemas para las redes LAN inalámbricas es **MACA** (**Acceso Múltiple con Prevención de Colisiones**, del inglés *Multiple Access with Collision Avoidance*) (Karn, 1990). El concepto en que se basa es que el emisor estimule al receptor para que envíe una trama corta, de manera que las estaciones cercanas puedan detectar esta transmisión y eviten ellas mismas hacerlo durante la siguiente trama de datos (grande). Se utiliza esta técnica en vez de la detección de portadora.

El MACA se ilustra en la figura 4-12. Consideremos ahora la manera en que *A* envía una trama a *B*. *A* comienza enviando una trama **RTS** (**Solicitud de Envío**, del inglés *Request To Send*) a *B*, como se muestra en la figura 4-12(a). Esta trama corta (30 bytes) contiene la longitud de la trama de datos que seguirá después. Después *B* contesta con una trama **CTS** (**Libre para Envío**, del inglés *Clear To Send*), como se muestra en la figura 4-12(b). La trama CTS contiene la longitud de los datos (que copia de la trama RTS). Al recibir la trama CTS, *A* comienza a transmitir.

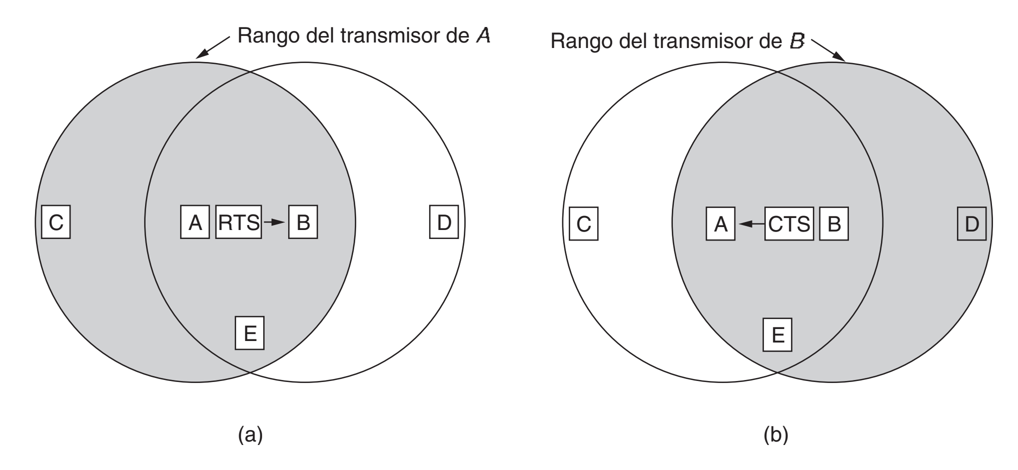


Figura 4-12. El protocolo MACA. (a) *A* envía un RTS a *B*. (b) *B* responde con un CTS a *A*.

Ahora veamos cómo reaccionan las estaciones que escuchan cualquiera de estas tramas. Sin duda, cualquier estación que escuche el RTS está bastante cerca de *A* y debe permanecer en silencio durante el tiempo suficiente para que el CTS se transmita de regreso a *A* sin conflicto. Es evidente que cualquier estación que escuche el CTS está bastante cerca de *B* y debe permanecer en silencio durante la siguiente transmisión de datos, cuya longitud puede determinar examinando la trama CTS.

En la figura 4-12, *C* está en el alcance de *A* pero no en el alcance de *B*. Por lo tanto, escucha el RTS de *A* pero no el CTS de *B*. En tanto no interfiera con el CTS, está libre para transmitir mientras se envía la trama de datos. En contraste, *D* está en el alcance de *B* pero no de *A*. No escucha el RTS pero sí el CTS. Al escuchar el CTS sabe que está cerca de una estación que está a punto de recibir una trama, por lo que difiere el envío de cualquier cosa hasta el momento en que se espera la terminación de esa trama. La estación *E* escucha ambos mensajes de control y, al igual que D, debe permanecer en silencio hasta que se haya completado la trama de datos.

A pesar de estas precauciones, aún pueden ocurrir colisiones. Por ejemplo, *B* y *C* podrían enviar tramas RTS a *A* al mismo tiempo. Éstas chocarán y se perderán. En el caso de una colisión, un transmisor sin éxito (es decir, uno que no escucha un CTS en el intervalo esperado) espera un tiempo aleatorio y vuelve a intentar más tarde.

4.3 ETHERNET

Hemos finalizado nuestra discusión general sobre los protocolos de asignación de canal, por lo que es tiempo de ver la forma en que estos principios se aplican a sistemas reales. Muchos de los diseños para las redes personales, locales y de área metropolitana se han estandarizado bajo el nombre de IEEE 802. Algunos han sobrevivido pero muchos no, como vimos en la figura 1-38. Quienes creen en la reencarna-

ción piensan que Charles Darwin regresó como miembro de la Asociación de estándares del IEEE para eliminar a los débiles. Los sobrevivientes más importantes son el 802.3 (Ethernet) y el 802.11 (LAN inalámbrica). Bluetooth (PAN inalámbrica) se utiliza mucho en la actualidad, pero se estandarizó fuera del 802.15. Todavía es muy pronto para decir algo sobre el 802.16 (MAN inalámbrica). Le sugerimos que consulte la sexta edición de este libro para averiguarlo.

Empezaremos nuestro estudio de los sistemas reales con Ethernet, que probablemente sea el tipo más ubicuo de red de computadoras en el mundo. Existen dos tipos de Ethernet: **Ethernet clásica**, que resuelve el problema de acceso múltiple mediante el uso de las técnicas que hemos estudiado en este capítulo; el segundo tipo es la **Ethernet conmutada**, en donde los dispositivos llamados switches se utilizan para conectar distintas computadoras. Es importante mencionar que, aunque se hace referencia a ambas como Ethernet, son muy diferentes. La Ethernet clásica es la forma original que operaba a tasas de transmisión de 3 a 10 Mbps. La Ethernet conmutada es en lo que se convirtió la Ethernet y opera a 100, 1 000 y 10 000 Mbps, en formas conocidas como Fast Ethernet, Gigabit Ethernet y 10 Gigabit Ethernet. Actualmente, en la práctica sólo se utiliza Ethernet conmutada.

Analizaremos estas formas históricas de Ethernet en orden cronológico para mostrar cómo se desarrollaron. Puesto que Ethernet y el IEEE 802.3 son idénticos, excepto por una pequeña diferencia (que veremos en breve), muchas personas usan los términos “Ethernet” e “IEEE 802.3” sin distinción. Nosotros también lo haremos. Para obtener más información sobre Ethernet, consulte a Spurgeon (2000).

4.3.1 Capa física de Ethernet clásica

La historia de Ethernet empieza casi al mismo tiempo que ALOHA, cuando un estudiante llamado Bob Metcalfe obtuvo su licenciatura en el MIT y después obtuvo su doctorado en Harvard. Durante sus estudios oyó hablar del trabajo de Abramson. Se interesó tanto en él que, después de graduarse de Harvard, decidió pasar el verano en Hawai trabajando con Abramson antes de empezar a trabajar en Xerox PARC (Palo Alto Research Center). Cuando llegó a PARC, vio que los investigadores ahí habían diseñado y construido lo que después se conocería como computadora personal. Pero las máquinas estaban aisladas. Haciendo uso de su conocimiento sobre el trabajo de Abramson, junto con su colega David Boggs diseñó e implementó la primera red de área local (Metcalfe y Boggs, 1976). Esta red utilizaba un solo cable coaxial grueso y extenso; operaba a 3 Mbps.

Llamaron al sistema **Ethernet** en honor al *éter luminífero*, por medio del cual se pensaba antes que se propagaba la radiación electromagnética (cuando el físico inglés del siglo XIX James Clerk Maxwell descubrió que la radiación electromagnética se podía describir mediante una ecuación de onda, los científicos asumieron que el espacio debía estar lleno de algún medio etéreo en el que se propagaba la radiación. No fue sino hasta después del famoso experimento de Michelson-Morley en 1887 que los físicos descubrieron que la radiación electromagnética se podía propagar en un vacío).

La Xerox Ethernet fue tan exitosa que DEC, Intel y Xerox idearon un estándar en 1978 para una Ethernet de 10 Mbps, conocido como **estándar DIX**. Con una modificación menor, el estándar DIX se convirtió en el estándar IEEE 802.3 en 1983. Por desgracia para Xerox, ya contaba con un historial de hacer inventos seminales (como la computadora personal) y después fracasar en su comercialización, una historia contada en la publicación *Fumbling the Future* (Smith y Alexander, 1988). Cuando Xerox mostró poco interés en hacer algo con Ethernet aparte de ayudar a estandarizarla, Metcalfe formó su propia empresa llamada 3Com para vender adaptadores de Ethernet para PC. Vendió muchos millones de ellos.

La Ethernet clásica se tendía alrededor del edificio como un solo cable largo al que se conectaban todas las computadoras. Esta arquitectura se muestra en la figura 4-13. La primera variedad, conocida popularmente como **Ethernet gruesa**, se asemejaba a una manguera de jardín amarilla, con marcas cada 2.5 metros para mostrar en dónde conectar las computadoras (el estándar 802.3 en realidad no *requería*

que el cable fuera amarillo, pero sí lo *sugería*). Después le siguió la **Ethernet delgada**, que se doblaba con más facilidad y las conexiones se realizaban mediante conectores BNC. La Ethernet delgada era mucho más económica y fácil de instalar, pero sólo se podían tender 185 metros por segmento (en vez de los 500 m con la Ethernet gruesa), cada uno de los cuales sólo podía manejar 30 máquinas (en vez de 100).

Cada versión de Ethernet tiene una longitud de cable máxima por segmento (es decir, longitud sin amplificar) a través de la cual se propagará la señal. Para permitir redes más grandes, se pueden conectar varios cables mediante **repetidores**. Un repetidor es un dispositivo de capa física que recibe, amplifica (es decir, regenera) y retransmite las señales en ambas direcciones. En cuanto a lo que al software concierne, una serie de segmentos de cable conectados por repetidores no presenta ninguna diferencia en comparación con un solo cable (excepto por una pequeña cantidad de retardo que introducen los repetidores).

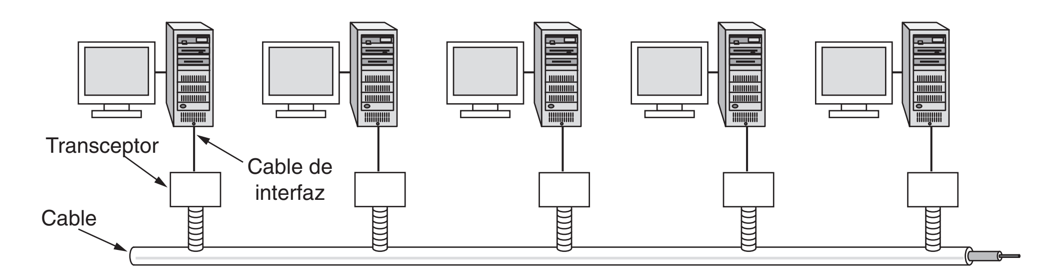


Figura 4-13. Arquitectura de Ethernet clásica.

La información se enviaba a través de cada uno de estos cables mediante la codificación Manchester que estudiamos en la sección 2.5. Una Ethernet podía contener varios segmentos de cable y múltiples repetidores, pero no podía haber dos transceptores separados por más de 2.5 km, y no podía haber una trayectoria entre dos transceptores en la que se colocaran más de cuatro repetidores. La razón de esta restricción era para que el protocolo MAC (que veremos a continuación) pudiera funcionar de manera correcta.

4.3.2 El protocolo de subcapa MAC de la Ethernet clásica

El formato utilizado para enviar tramas se muestra en la figura 4-14. Primero viene un *Preámbulo* de 8 bytes, cada uno de los cuales contiene el patrón de bits 10101010 (con la excepción del último byte, en el que los últimos 2 bits se establecen a 11). Este último byte se llama delimitador de *Inicio de trama* en el 802.3. La codificación de Manchester de este patrón produce una onda cuadrada de 10 MHz durante 6.4 μseg para permitir que el reloj del receptor se sincronice con el del emisor. Los últimos dos bits indican al receptor que está a punto de empezar el resto de la trama.

Bytes	8	6	6	2	0-1500	0-46	4
(a)	Preámbulo	Dirección de destino	Dirección de origen	Tipo	Datos	Relleno	Suma de verificación

(b)	Preámbulo	S o F	Dirección de destino	Dirección de origen	Longitud	Datos	Relleno	Suma de verificación

Figura 4-14. Formatos de trama. (a) Ethernet (DIX). (b) IEEE 802.3.

Después vienen dos direcciones, una para el destino y una para el origen. Cada una de ellas tiene una longitud de 6 bytes. El primer bit transmitido de la dirección de destino es un 0 para direcciones ordinarias y un 1 para direcciones de grupo. Las direcciones de grupo permiten que varias estaciones escuchen en una sola dirección. Cuando una trama se envía a una dirección de grupo, todas las estaciones del grupo la reciben. El envío a un grupo de estaciones se llama **multidifusión** (*multicasting*). La dirección especial que consiste únicamente en bits 1 está reservada para **difusión** (*broadcasting*). Una trama que contiene sólo bits 1 en el campo de destino se acepta en todas las estaciones de la red. La multidifusión es más selectiva, pero involucra el manejo de grupos para definir qué estaciones están en un grupo. Por el contrario, la difusión no hace ninguna diferencia entre las estaciones, por lo que no requiere manejo de grupos.

Una característica interesante de las direcciones de origen de las estaciones es que son globalmente únicas; el IEEE las asigna de manera central para asegurar que no haya dos estaciones en el mundo con la misma dirección. La idea es que cualquier estación pueda direccionar de manera exclusiva cualquier otra estación con sólo dar el número correcto de 48 bits. Para hacer esto, se utilizan los primeros 3 bytes del campo de dirección para un **OUI** (**Identificador Único Organizacional**, del inglés *Organizationally Unique Identifier*). El IEEE asigna los valores para este campo, e indican un fabricante. A los fabricantes se les asignan bloques de 2^{24} direcciones. El fabricante asigna los últimos 3 bytes de la dirección y programa la dirección completa en la NIC antes de venderla.

A continuación está el campo *Tipo* o *Longitud*, dependiendo de si la trama es Ethernet o IEEE 802.3. Ethernet usa un campo *Tipo* para indicar al receptor qué hacer con la trama. Es posible utilizar múltiples protocolos de capa de red al mismo tiempo en la misma máquina, por lo que cuando llega una trama de Ethernet, el sistema operativo tiene que saber a cuál entregarle la trama. El campo *Tipo* especifica a qué proceso darle la trama. Por ejemplo, un código de tipo de 0x0800 significa que los datos contienen un paquete IPv4.

Gracias a su sabiduría, el IEEE 802.3 decidió que este campo transportaría la longitud de la trama, ya que para determinar la longitud de Ethernet había que ver dentro de los datos; una violación del uso de capas, si alguna vez la hubo. Desde luego que esto significaba que no había forma de que el receptor averiguara qué hacer con una trama entrante. Para resolver ese problema se agregó otro encabezado para el protocolo **LLC** (**Control de Enlace Lógico**, del inglés *Logical Link Control*) dentro de los datos. Utiliza 8 bytes para transportar los 2 bytes de información del tipo del protocolo.

Por desgracia, para cuando se publicó el estándar 802.3, había ya tanto hardware y software para DIX Ethernet en uso que pocos fabricantes y usuarios se esforzaron en reempaquetar los campos *Tipo* y *Longitud*. En 1997, el IEEE desistió y dijo que estaba bien usar ambas formas. Por fortuna, todos los campos *Tipo* que se usaban antes de 1997 tenían valores mayores que 1500, que estaba bien establecido como el máximo tamaño de datos. Ahora la regla es que cualquier número ahí que sea menor o igual a 0x600 (1536) se puede interpretar como *Longitud*, y cualquier número mayor de 0x600 se puede interpretar como *Tipo*. Ahora el IEEE puede sostener que todos usan su estándar y que todos los demás pueden seguir haciendo lo que ya estaban haciendo (ignorar el LLC) sin sentirse culpables al respecto.

Después están los datos, de hasta 1500 bytes. Este límite fue elegido de manera algo arbitraria cuando se estableció el estándar Ethernet, sobre todo con base en el hecho de que un transceptor necesita suficiente RAM para mantener toda una trama y la RAM era muy costosa en 1978. Un mayor límite superior podría haber significado más RAM y, por ende, un transceptor más costoso.

Además de haber una longitud de trama máxima, también hay una longitud mínima. Si bien algunas veces un campo de datos de 0 bytes es útil, causa problemas. Cuando un transceptor detecta una colisión, trunca la trama actual, lo que significa que los bits perdidos y las piezas de las tramas aparecen todo el tiempo en el cable. Para que Ethernet pueda distinguir con facilidad las tramas válidas de lo inservible, necesita que dichas tramas tengan una longitud de por lo menos 64 bytes, de la dirección de destino a la suma de verificación, incluyendo ambas. Si la porción de datos de una trama es menor que 46 bytes, el campo de *Relleno* se utiliza para completar la trama al tamaño mínimo.

Otra razón (más importante) para tener una trama de longitud mínima es evitar que una estación complete la transmisión de una trama corta antes de que el primer bit llegue al extremo más alejado del cable, donde podría tener una colisión con otra trama. Este problema se ilustra en la figura 4-15. En el tiempo 0, la estación *A,* en un extremo de la red, envía una trama. Llamemos τ al tiempo que tarda en llegar esta trama al otro extremo. Justo antes de que la trama llegue al otro extremo (es decir, en el tiempo $\tau - \varepsilon$) la estación más distante, *B*, comienza a transmitir. Cuando *B* detecta que está recibiendo más potencia de la que está enviando, sabe que ha ocurrido una colisión, por lo que aborta su transmisión y genera una ráfaga de ruido de 48 bits para avisar a las demás estaciones. En otras palabras, bloquea el cable para asegurarse de que el emisor no ignore la colisión. Cerca del tiempo 2τ, el emisor ve la ráfaga de ruido y aborta también su transmisión. Luego espera un tiempo aleatorio antes de reintentarlo.

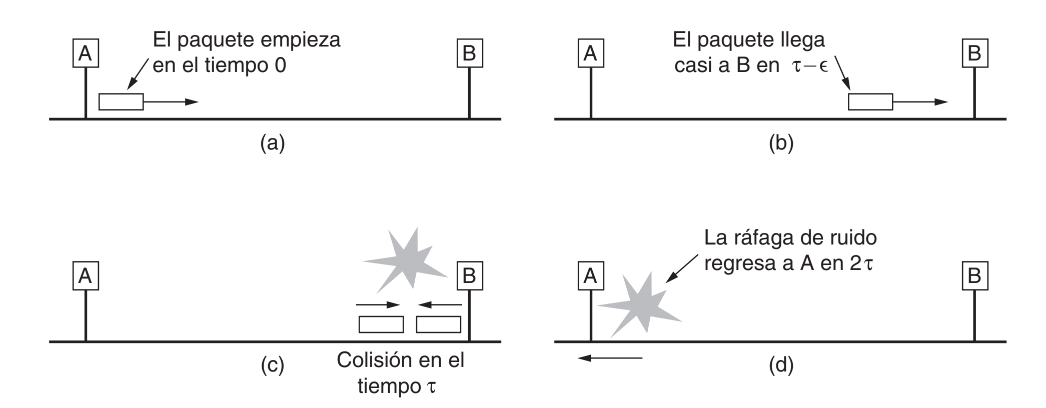


Figura 4-15. La detección de una colisión puede tardar hasta 2τ.

Si una estación intenta transmitir una trama muy corta, es concebible que ocurra una colisión, pero la transmisión se completará antes de que la ráfaga de ruido llegue de regreso a la estación en 2τ. El emisor entonces supondrá de forma incorrecta que la trama se envió con éxito. Para evitar que ocurra esta situación, todas las tramas deberán tardar más de 2τ para enviarse, de manera que la transmisión aún se esté llevando a cabo cuando la ráfaga de ruido regrese al emisor. Para una LAN de 10 Mbps con una longitud máxima de 2 500 metros y cuatro repetidores (de la especificación 802.3), el tiempo de ida y vuelta (incluyendo el tiempo de propagación a través de los cuatro repetidores) se ha determinado en cerca de 50 μseg en el peor de los casos. Por lo tanto, la trama más corta permitida se debe tardar por lo menos este tiempo en transmitir. A 10 Mbps, un bit tarda 100 nseg, por lo que 500 bits es la trama más pequeña que se garantiza funcionará. Para agregar algún margen de seguridad, este número se redondeó a 512 bits o 64 bytes.

El campo final de es la *Suma de verificación*. Es un CRC de 32 bits del tipo que estudiamos en la sección 3.2. De hecho, se define exactamente mediante el polinomio generador que vimos en esa sección, que funciona también para PPP, ADSL y otros enlaces. Esta CRC es un código de detección de errores que se utiliza para determinar si los bits de la trama se recibieron correctamente. Sólo realiza detección de errores y la trama se desecha si se detecta uno.

CSMA/CD con retroceso exponencial binario

La Ethernet clásica utiliza el algoritmo CSMA/CD persistente-1 que vimos en la sección 4.2. Este descriptor tan sólo significa que las estaciones detectan el medio cuando tienen una trama que desean enviar,

y la envían tan pronto como el medio está inactivo. Monitorean el canal por si hay colisiones al momento en que envían. Si hay una colisión, abortan la transmisión con una señal de bloqueo corta y vuelven a transmitir después de un intervalo aleatorio.

Ahora veamos cómo se determina el intervalo aleatorio cuando ocurre una colisión, ya que es un nuevo método. El modelo sigue siendo el de la figura 4-5. Tras una colisión, el tiempo se divide en ranuras discretas cuya longitud es igual al tiempo de propagación de ida y vuelta para el peor de los casos en el cable (2τ). Tomando en cuenta la ruta más larga permitida por Ethernet, el tiempo de ranura se estableció en 512 tiempos de bit o 51.2 μseg.

Después de la primera colisión, cada estación espera 0 o 1 tiempos de ranura al azar antes de intentarlo de nuevo. Si dos estaciones entran en colisión y ambas escogen el mismo número aleatorio, habrá una nueva colisión. Después de la segunda colisión, cada una escoge 0, 1, 2 o 3 al azar y espera ese tiempo de ranura. Si ocurre una tercera colisión (la probabilidad de que esto suceda es de 0.25), entonces para la siguiente vez el número de ranuras a esperar se escogerá al azar del intervalo 0 a $2^3 - 1$.

En general, después de i colisiones se elige un número aleatorio entre 0 y $2^i - 1$, y se salta ese número de ranuras. Sin embargo, al llegar a 10 colisiones el intervalo de aleatorización se congela en un máximo de 1 023 ranuras. Después de 16 colisiones, el controlador tira la toalla e informa a la computadora que fracasó. La recuperación posterior es responsabilidad de las capas superiores.

Este algoritmo, llamado **retroceso exponencial binario**, se escogió para adaptar en forma dinámica el número de estaciones que intentan transmitir. Si el intervalo de aleatorización para todas las colisiones fuera de 1023, la posibilidad de que chocaran dos estaciones una segunda vez sería insignificante, pero la espera promedio tras una colisión sería de cientos de tiempos de ranura, lo que introduce un retardo significativo. Por otra parte, si cada estación siempre se retardara 0 o 1 ranuras, entonces al tratar de transmitir 100 estaciones al mismo tiempo, habría colisiones una y otra vez hasta que 99 de ellas escogieran 1 y la estación restante escogiera 0. Esto podría tomar años. Al hacer que el intervalo de aleatorización crezca de manera exponencial a medida que ocurren cada vez más colisiones, el algoritmo asegura un retardo pequeño cuando sólo unas cuantas estaciones entran en colisión, pero también asegura que la colisión se resuelva en un intervalo razonable cuando haya colisiones entre muchas estaciones. Al truncar el retroceso a 1023, evitamos que el límite crezca demasiado.

Si no hay colisión, el emisor supone que la trama probablemente se entregó con éxito. Es decir, ni CSMA/CD ni Ethernet proveen confirmaciones de recepción. Esta elección es apropiada para los canales de cable de cobre y de fibra óptica que tienen tasas de error bajas. Cualquier error que ocurra debe entonces detectarse mediante la CRC y recuperarse en las capas superiores. Para los canales inalámbricos que tienen más errores, veremos que se utilizan confirmaciones de recepción.

4.3.3 Desempeño de Ethernet

Ahora examinaremos brevemente el desempeño de la Ethernet clásica en condiciones de carga pesada y constante; es decir, con k estaciones siempre listas para transmitir. Es complicado un análisis riguroso del algoritmo de retroceso exponencial binario. Sin embargo, seguiremos a Metcalfe y Boggs (1976) y supondremos una probabilidad constante de retransmisión en cada ranura. Si cada estación transmite durante una ranura de contención con una probabilidad p, la probabilidad A de que una estación adquiera el canal durante esa ranura es de:

$$A = kp(1 - p)^{k-1} \tag{4-5}$$

A se maximiza cuando $p = 1/k$, con $A \rightarrow 1/e$ conforme $k \rightarrow \infty$. La probabilidad de que el intervalo de contención tenga exactamente j ranuras es de $A(1 - A)^{j-1}$, por lo que el número medio de ranuras por contención está dado por

$$\sum_{j=0}^{\infty} jA(1-A)^{j-1} = \frac{1}{A}$$

Puesto que cada ranura tiene una duración de 2τ, el intervalo promedio de contención, w, es $2\tau/A$. Si suponemos una p óptima, el número promedio de ranuras de contención nunca es mayor que e, por lo que w es, cuando mucho, $2\tau e \approx 5.4\tau$.

Si la trama promedio tarda P segundos en transmitirse, cuando muchas estaciones tienen tramas por enviar,

$$\text{Eficiencia del canal} = \frac{P}{P + 2\tau/A} \tag{4-6}$$

Aquí vemos que la distancia máxima de cable entre dos estaciones entra en el cálculo de desempeño. Cuanto mayor sea la longitud del cable, mayor será el intervalo de contención, razón por la cual el estándar Ethernet especifica una longitud máxima de cable.

Es instructivo formular la ecuación (4-6) en términos de la longitud de trama, F, el ancho de banda de la red, B, la longitud del cable, L, y la velocidad de propagación de la señal, c, para el caso óptimo de e ranuras de contención por trama. Con $P = F/B$, la ecuación (4-6) se convierte en

$$\text{Eficiencia del canal} = \frac{1}{1 + 2BLe/cF} \tag{4-7}$$

Cuando el segundo término en el denominador sea grande, la eficiencia de la red será baja. Específicamente, un aumento en el ancho de banda o la distancia de la red (el producto BL) reduce la eficiencia para una trama de un tamaño dado. Por desgracia, mucha de la investigación sobre hardware de redes está enfocada a aumentar este producto. La gente quiere un gran ancho de banda a través de distancias grandes (por ejemplo, en las MAN de fibra óptica), lo que sugiere que tal vez la Ethernet clásica implementada de esta forma no sea el mejor sistema para estas aplicaciones. En la siguiente sección veremos otras formas de implementar Ethernet.

En la figura 4-16 se presenta una gráfica de la eficiencia del canal contra el número de estaciones listas para $2\tau = 51.2$ μseg y una tasa de transmisión de datos de 10 Mbps, usando la ecuación (4-7). Con un tiempo de ranura de 64 bytes, no es sorprendente que las tramas de 64 bytes no sean eficientes. Por otra parte, con tramas de 1 024 bytes y un valor asintótico de e ranuras de 64 bytes por intervalo de contención, el periodo de contención tiene 174 bytes de longitud y la eficiencia es del 85%. Este resultado es mucho mejor que la eficiencia de 37% del ALOHA ranurado.

Tal vez valga la pena mencionar que se ha realizado una gran cantidad de análisis teóricos del desempeño de Ethernet (y otras redes). Es conveniente tomar con cautela la mayoría de los resultados, por dos razones. Primero, casi todos estos trabajos han supuesto que el tráfico es Poisson. A medida que los investigadores han comenzado a examinar datos reales, se ha hecho evidente que el tráfico en redes pocas veces es Poisson. Al contrario, es autosimilar o de ráfaga a través de un rango de escalas de tiempo (Paxson y Floyd, 1995; Leland y colaboradores, 1994). Lo que esto significa es que el promedio durante periodos extensos no hace al tráfico más uniforme. Además de usar modelos cuestionables, muchos de los análisis se enfocan en los casos “interesantes” de desempeño con una carga inusualmente alta. Boggs y colaboradores (1988) mostraron mediante la experimentación que en realidad Ethernet funciona bien, incluso con una carga moderadamente alta.

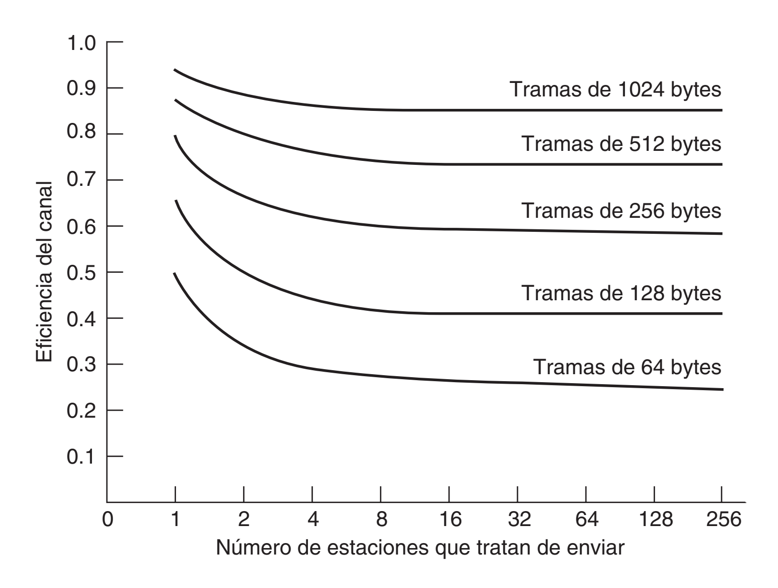


Figura 4-16. Eficiencia de Ethernet a 10 Mbps con tiempos de ranuras de 512 bits.

4.3.4 Ethernet conmutada

Pronto Ethernet empezó a evolucionar y a alejarse de la arquitectura de un solo cable extenso de la Ethernet clásica. Los problemas asociados con el hecho de encontrar interrupciones o conexiones flojas condujeron hacia un distinto tipo de patrón de cableado, en donde cada estación cuenta con un cable dedicado que llega a un **hub** (concentrador) central. Un hub simplemente conecta de manera eléctrica todos los cables que llegan a él, como si estuvieran soldados en conjunto. Esta configuración se muestra en la figura 4-17(a).

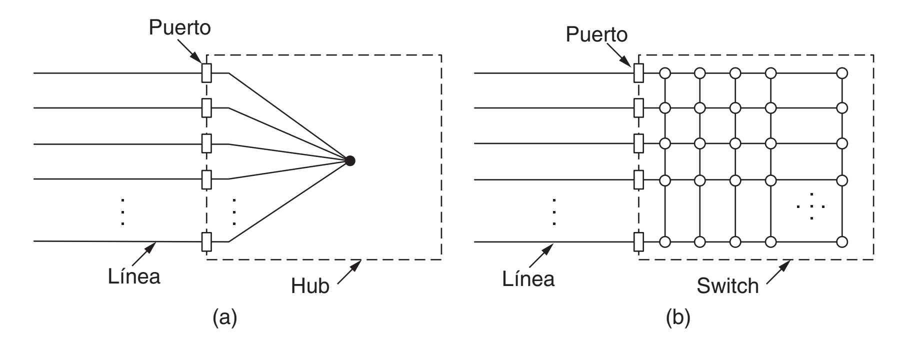


Figura 4-17. (a) Hub. (b) Switch.

Los cables eran pares trenzados de la compañía telefónica, ya que la mayoría de los edificios de oficinas contaban con este tipo de cableado y por lo general había muchos de sobra. Esta reutilización fue una ventaja, pero a la vez se redujo la distancia máxima de cable del hub hasta 100 metros (200 metros si se utilizaban pares trenzados categoría 5 de alta calidad). En esta configuración es más simple agregar o quitar una estación, además de que los cables rotos se pueden detectar con facilidad. Con las ventajas de usar el cableado existente y la facilidad de mantenimiento, los hubs de par trenzado se convirtieron rápidamente en la forma dominante de Ethernet.

Sin embargo, los hubs no incrementan la capacidad debido a que son lógicamente equivalentes al cable extenso individual de la Ethernet clásica. A medida que se agregan más estaciones, cada estación recibe una parte cada vez menor de la capacidad fija. En un momento dado, la LAN se saturará. Una forma de solucionar esto es usar una velocidad más alta; por decir, de 10 Mbps a 100 Mbp, 1 Gbps o incluso mayores velocidades. Pero con el crecimiento de multimedia y los poderosos servidores, incluso una Ethernet de 1 Gbps se podría saturar.

Por fortuna existe otra forma de tratar con el aumento de carga: una Ethernet conmutada. El corazón de este sistema es un **conmutador** (*switch*) que contiene un plano posterior (*backplane*) de alta velocidad, el cual conecta a todos los puertos como se muestra en la figura 4-17(b). Desde el exterior, un switch se ve igual que un hub. Ambos son cajas que por lo general contienen de 4 a 48 puertos, cada uno con un conector estándar RJ-45 r para un cable de par trenzado. Cada cable conecta al switch o hub con una sola computadora, como se muestra en la figura 4-18. Un switch tiene también las mismas ventajas que un hub. Es fácil agregar o quitar una nueva estación con sólo conectar o desconectar un cable, y es fácil encontrar la mayoría de las fallas, ya que un cable o puerto defectuoso por lo general afectará a una sola estación. De todas formas hay un componente compartido que puede fallar (el mismo switch), pero si todas las estaciones pierden conectividad, los encargados del TI sabrán qué hacer para corregir el problema: reemplazar el switch completo.

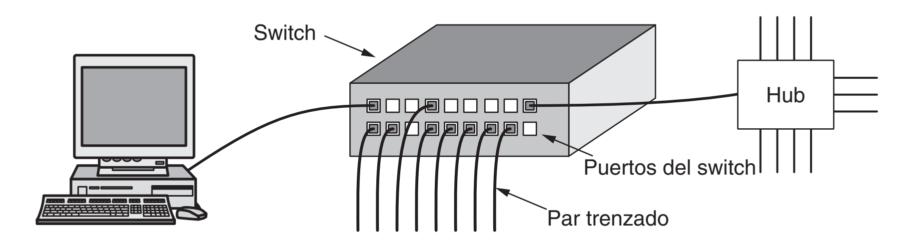


Figura 4-18. Un switch Ethernet.

Sin embargo, dentro del switch ocurre algo muy distinto. Los switches sólo envían tramas a los puertos para los cuales están destinadas. Cuando el puerto de un switch recibe una trama Ethernet de una estación, el switch verifica las direcciones de Ethernet para ver cuál es el puerto de destino de la trama. Este paso requiere que el switch sea capaz de deducir qué puertos corresponden a qué direcciones, un proceso que describiremos en la sección 4.8 cuando veamos el caso general de switches conectados a otros switches. Por ahora basta con suponer que el switch conoce el puerto de destino de la trama. A continuación, el switch reenvía la trama a través de su plano posterior de alta velocidad hacia el puerto de destino. Por lo general, el plano posterior opera a muchos Gbps mediante el uso de un protocolo propietario que no necesita estandarización, ya que está completamente oculto dentro del switch. Después, el puerto de destino transmite la trama sobre el cable, de manera que pueda llegar a la estación de destino. Ninguno de los otros puertos sabe siquiera que existe la trama.

¿Qué ocurre si más de una estación o puerto desea enviar una trama al mismo tiempo? De nuevo, los switches difieren de los hubs. En un hub, todas las estaciones están en el mismo **dominio de colisión**. Deben usar el algoritmo CSMA/CD para programar sus transmisiones. En un switch, cada puerto es su propio dominio de colisión independiente. En el caso común en que el cable es full-dúplex, tanto la estación como el puerto pueden enviar una trama en el cable al mismo tiempo, sin preocuparse por los demás puertos y estaciones. Ahora las colisiones son imposibles y no se necesita CSMA/CD. Pero si el cable es half-dúplex, la estación y el puerto deben competir por la transmisión con CSMA/CD de la manera usual.

Un switch mejora el desempeño de la red en comparación con un hub de dos maneras. Primero, como no hay colisiones, la capacidad se utiliza con más eficiencia. Segundo y más importante, con un switch se pueden enviar varias tramas al mismo tiempo (por distintas estaciones). Estas tramas llegarán a los puertos del switch y viajarán hacia el plano posterior de éste para enviarlos por los puertos apropiados. No obstante, como se podrían enviar dos tramas al mismo puerto de salida y al mismo tiempo, el switch debe tener un búfer para que pueda poner temporalmente en cola una trama de entrada hasta que se pueda transmitir al puerto de salida. En general, estas mejoras producen una considerable ganancia en el desempeño que no es posible lograr con un hub. Con frecuencia, la velocidad real de transmisión total del sistema se puede incrementar en un orden de magnitud, dependiendo del número de puertos y patrones de tráfico.

El cambio en los puertos por donde se envían las tramas también incluye beneficios de seguridad. La mayoría de las interfaces de LAN tienen un **modo promiscuo**, en el que *todas* las tramas se entregan a cada computadora y no sólo las que van dirigidas a ella. En un hub, cualquier computadora conectada puede ver el tráfico transmitido entre todas las demás computadoras. Los espías y los intrusos aman esta característica. En un switch, el tráfico se reenvía sólo a los puertos a los que está destinado. Esta restricción provee un mejor aislamiento, de modo que el tráfico no escape fácilmente y caiga en las manos equivocadas. Sin embargo, es mejor cifrar el tráfico si de verdad se necesita seguridad.

Como el switch sólo espera tramas Ethernet estándar en cada puerto de entrada, es posible usar algunos de los puertos como concentradores. En la figura 4-18, el puerto en la esquina superior derecha no está conectado a una sola estación, sino a un hub de 12 puertos. A medida que llegan tramas al hub, compiten por el cable de la manera usual, incluyendo las colisiones y el retroceso binario. Las tramas que tienen éxito pasan por el hub hasta el switch, en donde se tratan como cualquier otra trama entrante. El switch no sabe que tuvieron que competir para entrar. Una vez en el switch, se envían a la línea de salida correcta a través del plano posterior de alta velocidad. También es posible que el destino correcto estuviera en una de las líneas conectadas al hub, en cuyo caso la trama ya se entregó y el switch simplemente la descarta. Los hubs son más simples y económicos que los switches, pero debido a que estos últimos han reducido su precio constantemente, los primeros se han convertido en una especie en extinción. Las redes modernas usan en su mayor parte Ethernet conmutada. Sin embargo, aún existen los hubs heredados.

4.3.5 Fast Ethernet

Al mismo tiempo que los switches ganaban popularidad, la velocidad de 10 Mbps de Ethernet estaba bajo una presión cada vez mayor. Al principio, 10 Mbps parecían el cielo, al igual que los módems de cable parecieron el cielo a los usuarios de los módems telefónicos. Pero la novedad desapareció muy rápido. Como un tipo de corolario a la Ley de Parkinson ("El trabajo se expande hasta llenar el tiempo disponible para que se termine"), tal pareciera que los datos se expandieron hasta llenar el ancho de banda disponible para su transmisión.

Muchas instalaciones necesitaban más ancho de banda y, por lo tanto, tenían numerosas redes LAN de 10 Mbps conectadas por una maraña de repetidores, hubs y switches, aunque los administradores de redes algunas veces sentían que las conexiones parecían estar hechas con goma de mascar y tela metálica. Pero incluso con los switches de Ethernet, el ancho de banda máximo de una sola computadora estaba limitado por el cable que lo conectaba con el puerto del switch.

Fue en este entorno que el IEEE convocó al comité 802.3 en 1992 con instrucciones de idear una LAN más rápida. Una propuesta fue mantener la red 802.3 igual a como estaba, sólo que hacerla más rápida. Otra propuesta fue rehacerla en su totalidad para darle muchas características nuevas, como tráfico en tiempo real y voz digitalizada, pero mantener el nombre antiguo (por razones de marketing). Después

de algunas discusiones, el comité decidió mantener la Ethernet 802.3 tal como estaba, pero hacerla más rápida. Esta estrategia cumpliría con el objetivo antes de que cambiara la tecnología, además de evitar los problemas imprevistos con un diseño totalmente nuevo. El nuevo diseño también sería compatible con las versiones previas de redes LAN Ethernet existentes. Las personas que apoyaban la propuesta contraria hicieron lo que cualquier persona de la industria de la computación habría hecho bajo estas circunstancias: unieron fuerzas, formaron su propio comité y estandarizaron su LAN de todas maneras (que con el tiempo se llamó 802.12). Pero fracasó rotundamente.

El trabajo se terminó muy rápido (mediante las normas de los comités de estándares) y el resultado, 802.3u, fue aprobado de manera oficial por el IEEE en junio de 1995. Técnicamente, 802.3u no es un nuevo estándar sino un agregado al estándar, 802.3 existente (para enfatizar su compatibilidad con versiones anteriores). Esta estrategia es muy utilizada. Puesto que prácticamente todos lo llaman **Fast Ethernet** en vez de 802.3u. Nosotros también lo haremos.

La idea básica detrás de Fast Ethernet era simple: mantener todos los formatos, interfaces y reglas de procedimientos anteriores, pero reducir el tiempo de bits de 100 nseg a 10 nseg. Técnicamente, habría sido posible copiar la Ethernet clásica de 10 Mbps y aún detectar colisiones a tiempo con sólo reducir la longitud máxima de cable por un factor de 10. Sin embargo, las ventajas del cableado de par trenzado eran tan abrumadoras que Fast Ethernet se basa por completo en este diseño. Por lo tanto, todos los sistemas Fast Ethernet utilizan hubs y switches; no se permiten cables con múltiples derivaciones vampiro ni conectores BNC.

Sin embargo, aún había que tomar algunas decisiones, siendo la más importante de todas qué tipos de cable soportar. Una opción era el cable de par trenzado categoría 3. El argumento a su favor era que casi todas las oficinas en el mundo occidental tenían por lo menos cuatro cables de par trenzado categoría 3 (o mejor) que iban desde ahí hasta un gabinete de cableado telefónico dentro de una distancia de 100 metros. Algunas veces había dos de esos cables. Por lo tanto, al usar cable de par trenzado categoría 3 se podrían cablear las computadoras de escritorio mediante Fast Ethernet sin tener que volver a cablear el edificio, lo cual es una enorme ventaja para muchas organizaciones.

La principal desventaja del cable de par trenzado categoría 3 es su incapacidad de transportar 100 Mbps a más de 100 metros, la máxima distancia de computadora a hub especificada para hubs de 10 Mbps. En contraste, el cable de par trenzado categoría 5 puede manejar 100 metros con facilidad, y la fibra puede recorrer mucha más distancia. El compromiso elegido fue permitir las tres posibilidades, como se muestra en la figura 4-19, pero fortalecer la solución categoría 3 para darle la capacidad de transmisión adicional necesaria.

Nombre	Cable	Segmento máximo	Ventajas
100Base-T4	Par trenzado	100 m	Utiliza UTP categoría 3.
100Base-TX	Par trenzado	100 m	Full-dúplex a 100 Mbps (UTP cat 5).
100Base-FX	Fibra óptica	2000 m	Full-dúplex a 100 Mbps; distancias largas.

Figura 4-19. El cableado original de Fast Ethernet.

El esquema UTP categoría 3, llamado **100Base-T4**, utilizaba una velocidad de señalización de 25 MHz, tan sólo un 25% más rápida que los 20 MHz de la Ethernet estándar (recuerde que la codificación Manchester, que vimos en la sección 2.5, requiere dos periodos de reloj para cada uno de los 10 millones de bits que se envían cada segundo). Sin embargo, para alcanzar la tasa de bits necesaria, 100Base-T4 requiere cuatro cables de par trenzado. De los cuatro pares, uno siempre va al hub, uno siempre sale del hub y los otros

dos se pueden conmutar a la dirección actual de la transmisión. Para obtener 100 Mbps de los tres pares trenzados en la dirección de la transmisión, se utiliza un esquema bastante complejo en cada par trenzado, que implica enviar dígitos ternarios con tres distintos niveles de voltaje. Es poco probable que este esquema vaya a ganar premios por su elegancia, por lo que omitiremos los detalles. Sin embargo, y como el cableado telefónico estándar ha tenido durante décadas cuatro pares trenzados por cable, la mayoría de las oficinas pueden usar la planta de cableado existente. Claro que esto significa renunciar al teléfono de su oficina, pero sin duda es un pequeño precio a pagar para obtener correo electrónico, que es más rápido.

100Base-T4 quedó al borde del camino debido a que se actualizó el cableado de muchos edificios de oficinas por UTP categoría 5 para Ethernet **100Base-TX**, el cual llegó a dominar el mercado. Este diseño es más simple puesto que los cables pueden manejar velocidades de reloj de 125 MHz. Sólo se utilizan dos pares trenzados por estación, uno que va al hub y otro que viene de él. No se utiliza la codificación binaria directa (es decir, NRZ) ni la codificación Manchester. En cambio se utiliza la codificación **4B/5B** que describimos en la sección 2.5. Se codifican 4 bits de datos como 5 bits de señal y se envían a 125 MHz para proveer 100 Mbps. Este esquema es simple pero tiene suficientes transiciones para la sincronización, además de que utiliza muy bien el ancho de banda del cable. El sistema 100Base-TX es full-dúplex; las estaciones pueden transmitir a 100 Mbps en un par trenzado y recibir a 100 Mbps en otro par trenzado al mismo tiempo.

La última opción, **100Base-FX**, utiliza dos filamentos de fibra multimodo, una para cada dirección, por lo que también es full-dúplex con 100 Mbps en cada dirección. En esta configuración, la distancia entre una estación y el switch puede ser de hasta 2 km.

Fast Ethernet permite la interconexión mediante hubs o switches. Para asegurar que el algoritmo CSMA/CD siga trabajando, es necesario mantener la relación entre el tamaño mínimo de trama y la longitud máxima del cable a medida que la velocidad de la red aumenta de 10 Mbps a 100 Mbps. Así, el tamaño mínimo de trama de 64 bytes debe aumentar o la longitud máxima de cable de 2 500 debe disminuir, en forma proporcional. La elección fácil fue reducir la distancia máxima entre dos estaciones cualesquiera por un factor de 10, puesto que un hub con cables de 100 m ya se encuentra dentro de este nuevo valor máximo. Sin embargo, los cables 100Base-FX de 2 km son demasiado largos como para permitir un hub de 100 Mbps con el algoritmo de colisiones normal de Ethernet. Estos cables se deben conectar a un switch y operar en un modo full-dúplex para que no haya colisiones.

Los usuarios empezaron a implementar con rapidez el estándar Fast Ethernet, pero no deseaban tirar las tarjetas Ethernet de 10 Mbps en las computadoras antiguas. Como consecuencia, casi todos los switches Fast Ethernet pueden manejar una mezcla de estaciones de 10 Mbps y 100 Mbps. Para facilitar la actualización, el estándar provee por sí solo un mecanismo llamado **autonegociación**, el cual permite que dos estaciones negocien de manera automática la velocidad óptima (10 o 100 Mbps) y la duplicidad (half-dúplex o full-dúplex). Funciona bien la mayor parte del tiempo, pero se sabe que provoca problemas de desajuste de duplicidad cuando un extremo del enlace realiza la autonegociación pero el otro extremo no, y se establece en modo full-dúplex (Shalunov y Carlson, 2005). La mayoría de los productos Ethernet usan esta característica para configurarse a sí mismos.

4.3.6 Gigabit Ethernet

La tinta apenas se estaba secando en el estándar de la Fast Ethernet cuando el comité 802 comenzó a trabajar en una Ethernet aún más rápida, que de inmediato recibió el apodo de **Gigabit Ethernet**. El IEEE ratificó la forma más popular como 80.3ab en 1999. A continuación analizaremos algunas de las características clave de la Gigabit Ethernet. Para mayor información consulte a Spurgeon (2000).

Los objetivos del comité para la Gigabit Ethernet eran en esencia los mismos que los del comité para Fast Ethernet: que tuviera un desempeño 10 veces mayor y que mantuviera la compatibilidad con

todos los estándares Ethernet existentes. En particular, Gigabit Ethernet tenía que ofrecer servicio de datagramas sin confirmación de recepción con unidifusión y multidifusión, utilizar el mismo esquema de direccionamiento de 48 bits que ya estaba en uso y mantener el mismo formato de trama, incluyendo los tamaños mínimo y máximo de trama. El estándar final cumple con todos estos objetivos.

Al igual que Fast Ethernet, todas las configuraciones de Gigabit Ethernet usan enlaces punto a punto. En la configuración más simple, que se muestra en la figura 4-20(a), dos computadoras están conectadas directamente una con otra. Sin embargo, el caso más común es tener un switch o un hub conectado a varias computadoras y quizás a switches o hubs adicionales, como se muestra en la figura 4-20(b). En ambas configuraciones, cada cable Ethernet individual tiene exactamente dos dispositivos en él, ni más ni menos.

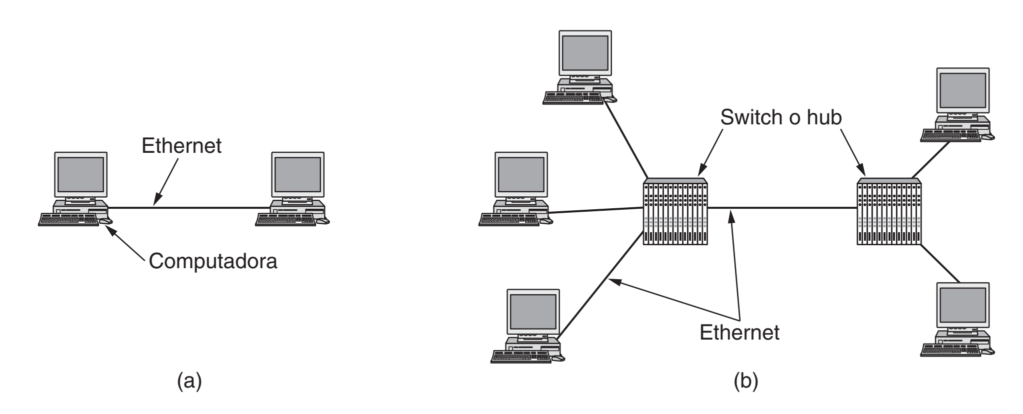


Figura 2-20. (a) Ethernet de dos estaciones. (b) Ethernet multiestación.

Además, al igual que Fast Ethernet, Gigabit Ethernet soporta dos modos diferentes de funcionamiento: modo full-dúplex y modo half-dúplex. El modo "normal" es el modo full-dúplex, que permite tráfico en ambas direcciones al mismo tiempo. Este modo se utiliza cuando hay un switch central conectado a computadoras (o a otros switches) en la periferia. En esta configuración, todas las líneas se almacenan en el búfer con el fin de que cada computadora y switch pueda enviar tramas siempre que lo desee. El emisor no tiene que detectar el canal para ver si alguien más lo está utilizando debido a que la contención es imposible. En la línea entre una computadora y un switch, la computadora es la única que puede enviar al switch y la transmisión tendrá éxito aun cuando el switch esté enviado ahora una trama a la computadora (porque la línea es full-dúplex). Debido a que no hay contención, no se utiliza el protocolo CSMA/CD y la longitud máxima del cable se determina con base en los aspectos relacionados con la fuerza de la señal, en vez de basarse en el tiempo que tarda una ráfaga de ruido en propagarse de vuelta al emisor en el peor de los casos. Los switches tienen la libertad de mezclar e igualar velocidades. La autonegociación se soporta al igual que en Fast Ethernet, sólo que ahora la opción está entre 10, 100 y 1000 Mbps.

El otro modo de operación es half-dúplex y se utiliza cuando las computadoras están conectadas a un hub en vez de un switch. Un hub no almacena las tramas entrantes. En su lugar, conecta en forma eléctrica todas las líneas internamente, simulando el cable con múltiples derivaciones que se utiliza en la Ethernet clásica. En este modo puede haber colisiones, por lo que se requiere el protocolo CSMA/CD estándar. Debido a que ahora se puede transmitir una trama de 64 bytes (la más corta permitida) 100 veces más rápido que en la Ethernet clásica, la longitud máxima del cable debe ser 100 veces menor, o de 25 metros, para mantener la propiedad esencial de que el emisor aún transmita cuando la ráfaga de ruido regrese a él, incluso en el peor de los casos. Con un cable de 2 500 metros de longitud, el emisor de una trama de 64 bytes

a 1 Gbps podría terminar su transmisión mucho antes de que la trama recorriera siquiera una décima del camino, y por ende muchísimo antes de que llegara al otro extremo y regresara.

Esta restricción de longitud fue tan dolorosa que se agregaron dos características al estándar para incrementar la longitud máxima del cable a 200 metros, lo cual quizás es suficiente para la mayoría de las oficinas. La primera característica, llamada **extensión de portadora**, básicamente indica al hardware que agregue su propio relleno después de la trama normal para extenderla a 512 bytes. Como el hardware emisor agrega este relleno y el hardware receptor lo elimina, el software no toma parte en esto, lo que significa que no es necesario realizar cambios al software existente. La desventaja es que usar 512 bytes de ancho de banda para transmitir 46 bytes de datos de usuario (la carga útil de una trama de 64 bytes) tiene una eficiencia de sólo un 9%.

La segunda característica, llamada **ráfagas de trama**, permite que un emisor transmita una secuencia concatenada de múltiples tramas en una sola transmisión. Si la ráfaga total es menor que 512 bytes, el hardware la rellena de nuevo. Si hay suficientes tramas esperando su transmisión, este esquema es muy eficiente y se prefiere en vez de la extensión de portadora.

Sin embargo, haciendo justicia, es difícil imaginar a una organización comprando computadoras modernas con tarjetas Gigabit Ethernet y después conectarlas con un hub anticuado para simular la Ethernet clásica con todas sus colisiones. Las interfaces y los switches de Gigabit Ethernet solían ser costosos, pero sus precios cayeron rápidamente cuando empezaron a aumentar los volúmenes de ventas. Aún así, la compatibilidad con versiones anteriores es algo sagrado en la industria de las computadoras, por lo que el comité tuvo que añadirla. En la actualidad, la mayoría de las computadoras incluyen una interfaz Ethernet capaz de operar a 10, 100 y 1000 Mbps, además de ser compatible con todas estas redes.

Como se lista en la figura 4-21, Gigabit Ethernet soporta tanto el cableado de cobre como el de fibra óptica. La señalización en, o cerca de, 1 Gbps requiere codificar y enviar un bit cada nanosegundo. En un principio este truco se lograba con cables cortos de cobre blindados (la versión 1000Base-CX) y con fibra óptica. En la fibra óptica se permiten dos longitudes de onda y resultan dos versiones distintas: 0.85 micras (corto, para 1000Base-SX) y 1.3 micras (largo, para 1000Base-LX).

Nombre	Cable	Segmento máximo	Ventajas
1000Base-SX	Fibra óptica	550 m	Fibra multimodo (50, 62.5 micras)
1000Base-LX	Fibra óptica	5000 m	Monomodo (10 µ) o multimodo (50, 62.5µ)
1000Base-CX	2 pares de STP	25 m	Par trenzado blindado
1000Base-T	4 pares de UTP	100 m	UTP estándar categoría 5

Figura 4-21. Cableado de Gigabit Ethernet.

La señalización en la longitud de onda corta se puede realizar mediante LEDs económicos. Se utiliza con fibra multimodo y es útil para las conexiones dentro de un edificio, ya que puede funcionar hasta por 500 m para la fibra de 50 micras. La señalización en la longitud de onda larga requiere láser más costosos. Por otro lado, al combinarse con fibra monomodo (10 micras), la longitud de cable puede ser de hasta 5 km. Este límite permite conexiones de larga distancia entre edificios (por ejemplo, la red troncal de un campus) como un enlace punto a punto dedicado. Las versiones posteriores del estándar permitían enlaces aún más largos a través de fibra monomodo.

Para enviar bits por estas versiones de Gigabit Ethernet, se pidió prestada la codificación **8B/10B** que describimos en la sección 2.5 de otra tecnología de red, conocida como **Canal de fibra**. Este esquema codifica 8 bits de datos en palabras codificadas de 10 bits que se envían a través del cable o la fibra, de

aquí que se llame 8B/10B. Las palabras codificadas se eligieron de modo que se pudieran balancear (es decir, que tengan el mismo número de 0s y 1s) con suficientes transiciones para la recuperación del reloj. Para enviar los bits codificados con NRZ se requiere un ancho de banda de señalización de un 25% más que el requerido para los bits sin codificación, una importante mejora en comparación con la expansión de 100% de la codificación Manchester.

Sin embargo, todas estas opciones requerían nuevos cables de cobre o de fibra para soportar la señalización más rápida. Ninguna de ellas hizo uso de la gran cantidad de cable UTP categoría 5 que se había instalado con la red Fast Ethernet. Antes de un año llegó 1000Base-T para llenar este vacío, y desde entonces ha sido la forma más popular de Gigabit Ethernet. Aparentemente a las personas no les gustaba tener que volver a cablear sus edificios.

Para hacer que Ethernet opere a 1000 Mbps a través de cables categoría 5 se necesita una señalización más complicada. Para empezar se utilizan los cuatro pares trenzados en el cable, y cada par se utiliza en ambas direcciones al mismo tiempo mediante el uso de un procesamiento de señales digitales para separar las señales. En cada cable se utilizan cinco niveles de voltaje que transportan 2 bits para una señalización de 125 Msímbolos/seg. La asignación para producir los símbolos a partir de los bits no es simple. Se requiere un mezclado (*scrambling*) para las transiciones, seguido de un código de corrección de errores en el que se incrustan cuatro valores en cinco niveles de señal.

1 Gbps es una velocidad muy alta. Por ejemplo, si un receptor está ocupado con otra tarea por incluso un 1 mseg y no vacía el búfer de entrada en alguna línea, podrían haberse acumulado hasta 1 953 tramas en ese espacio. Además, cuando una computadora en una Gigabit Ethernet está enviando datos por la línea a una computadora en una Ethernet clásica, es muy probable que sucedan desbordamientos de búfer. Como consecuencia de estas dos observaciones, Gigabit Ethernet soporta el control de flujo. El mecanismo consiste en que un extremo envíe una trama de control especial al otro extremo para indicarle que se detenga durante cierto periodo. Estas tramas de control PAUSE son tramas Ethernet comunes que contienen un tipo de 0x8808. Las pausas se dan en unidades del tiempo mínimo de trama. Para Gigabit Ethernet, la unidad de tiempo es de 512 nseg y se permiten pausas hasta de 33.6 mseg.

Hay una extensión más que se introdujo junto con Gigabit Ethernet. Las **tramas Jumbo** que permiten que las tramas tengan una longitud mayor de 1500 bytes, por lo general de hasta 9 KB. Esta extensión es propietaria. No la reconoce el estándar debido a que si se utiliza, entonces Ethernet ya no es compatible con las versiones anteriores, pero la mayoría de los distribuidores la soporta de todas formas. La razón es que 1500 bytes es una unidad corta a velocidades de gigabits. Al manipular bloques más grandes de información, la tasa de transmisión de tramas se puede reducir, junto con el procesamiento asociado con ésta, tal como interrumpir al procesador para decir que llegó una trama, o dividir y recombinar mensajes que eran demasiado largos como para caber en una trama Ethernet.

4.3.7 10 Gigabit Ethernet

Tan pronto como el Gigabit Ethernet se estandarizó, el comité 802 se aburrió y quiso volver al trabajo. El IEEE les dijo que iniciaran una Ethernet de 10 gigabits. Este trabajo siguió casi el mismo patrón que los estándares Ethernet anteriores, en donde aparecieron estándares para fibra y cable de cobre blindado por primera vez en 2002 y 2004, seguidos de un estándar para par trenzado de cobre en 2006.

10 Gbps es una velocidad realmente prodigiosa, 1 000 veces más rápida que la Ethernet original. ¿En dónde se podría necesitar? La respuesta es que dentro de los centros e intercambios de datos para conectar enrutadores, switches y servidores de gama alta, así como en las troncales de larga distancia con alto ancho de banda entre las oficinas que permiten la operación de redes de área metropolitana completas, basadas en Ethernet y fibra. Las conexiones de larga distancia usan fibra óptica, mientras que las conexiones cortas pueden usar cobre o fibra.

Todas las versiones de Ethernet de 10 gigabits soportan sólo la operación full-dúplex. CSMA/CD ya no forma parte del diseño y los estándares se concentran en los detalles de las capas físicas que pueden operar a muy alta velocidad. Pero la compatibilidad aún sigue siendo importante, por lo que las interfaces Ethernet de 10 gigabits usan la autonegociación y cambian a la velocidad más alta soportada por ambos extremos de la línea.

En la figura 4-22 se listan los principales tipos de Ethernet de 10 gigabits. Se utiliza fibra multimodo con la longitud de onda de 0.85 μ (corta) para distancias medias, y la fibra monomodo a 1.3 μ (larga) y 1.5 μ (extendida) para distancias largas. 10GBase-ER puede operar en distancias de 40 km, lo cual la hace adecuada para aplicaciones de área amplia. Todas estas versiones envían un flujo serial de información que se produce mediante el mezclado de los bits de datos, para después codificarlos mediante un código **64B/66B**. Esta codificación tiene menos sobrecarga que un código 8B/10B.

Nombre	Cable	Segmento máximo	Ventajas
10GBase-SR	Fibra óptica	Hasta 300 m	Fibra multimodo (0.85 μ).
10GBase-LR	Fibra óptica	10 km	Fibra monomodo (1.3 μ).
10GBase-ER	Fibra óptica	40 km	Fibra monomodo (1.5 μ).
10GBase-CX4	4 pares de twinax	15 m	Cobre twinaxial.
10GBase-T	4 pares de UTP	100 m	UTP categoría 6a

Figura 4-22. Cableado de Ethernet de 10 gigabits.

La primera versión de cobre que se definió (10GBase-CX4) utiliza un cable con cuatro pares de cableado twinaxial de cobre. Cada par usa codificación 8B/10B y opera a 3.125 Gsímbolos/segundo para alcanzar 10 Gbps. Esta versión es más económica que la fibra y fue de las primeras en comercializarse, pero aún está por verse si será vencida a la larga por la Ethernet de 10 gigabits sobre la variedad más común de cableado de par trenzado.

10GBase-T es la versión que usa cables UTP. Aunque requiere cableado categoría 6a, en distancias más cortas puede usar categorías más bajas (incluyendo la categoría 5) para reutilizar una parte del cableado ya instalado. No es sorpresa que la capa física esté bastante involucrada para llegar a 10 Gbps sobre par trenzado. Sólo esbozaremos algunos de los detalles de alto nivel. Cada uno de los cuatro pares trenzados se utiliza para enviar 2 500 Mbps en ambas direcciones. Para llegar a esta velocidad se utiliza una tasa de señalización de 800 Msímbolos/seg, con símbolos que usan 16 niveles de voltaje. Para producir los símbolos se mezclan los datos, se protegen con un código LDPC (Verificación de Paridad de Baja Densidad, del inglés *Low Density Parity Check*) y se vuelven a codificar para corrección de errores.

La Ethernet de 10 gigabits se sigue tambaleando en el mercado, pero el comité 802.3 ya siguió adelante. A finales de 2007, el IEEE creó un grupo para estandarizar la Ethernet que opera a 40 Gbps y 100 Gbps. Esta actualización permitirá a Ethernet competir en ambientes de muy alto rendimiento, incluyendo las conexiones de larga distancia en redes troncales y las conexiones cortas a través de los planos posteriores de los equipos. El estándar todavía no está completo, pero ya hay productos propietarios disponibles.

4.3.8 Retrospectiva de Ethernet

Ethernet ha existido desde hace casi 30 años y no tiene competidores serios a la vista, por lo que es probable que exista por algunos años más. Pocas arquitecturas de CPU, sistemas operativos o lenguajes

de programación han sido los reyes de la montaña por tres décadas sin mostrar signos de debilidad. Es evidente que Ethernet hizo algo bien, pero, ¿qué fue?

Quíza la razón principal de su longevidad es que Ethernet es simple y flexible. En la práctica, simple se traduce como confiable, económico y fácil de mantener. Una vez que se adoptó la arquitectura de hub y switch, casi no ocurrían fallas. Las personas dudaban en reemplazar algo que funciona bien todo el tiempo, sobre todo cuando saben que muchas cosas funcionan pobremente en la industria de la computación, de modo que muchas de las denominadas "actualizaciones" son peores que lo que reemplazan.

Simple también se traduce como económico. El cableado de par trenzado tiene un costo relativamente bajo, al igual que los componentes de hardware. Pueden empezar con un alto costo cuando hay una transición; por ejemplo, nuevas NICs o switches de Gigabit Ethernet, pero son simples adiciones para una red bien establecida (no su reemplazo) y los precios bajan con rapidez a medida que el volumen de ventas aumenta.

Ethernet es fácil de mantener. No hay software que instalar (sólo los controladores) y tampoco hay tablas de configuración que manejar (con el riesgo de equivocarse). Además, agregar nuevos hosts es tan simple como conectarlos.

Otro punto es que Ethernet interactúa fácilmente con TCP/IP, el cual se ha vuelto dominante. IP es un protocolo sin conexión, por lo que se adapta muy bien a Ethernet, que también es sin conexión. IP no se adapta tan bien a las alternativas orientadas a conexión como ATM. Esta falta de ajuste afecta definitivamente las posibilidades de ATM.

Por último, y tal vez lo más importante, Ethernet ha sido capaz de evolucionar en ciertas formas importantes. Las velocidades han aumentado en varios órdenes de magnitud y se han introducido los hubs y switches, pero estos cambios no requieren modificaciones en el software, lo cual ha permitido que se reutilice muchos el cableado existente durante un tiempo. Si un vendedor de redes aparece en una instalación grande y dice: "Tengo esta nueva red fantástica para usted. Lo único que tiene que hacer es tirar todo su hardware y reescribir todo su software", está en un grave problema.

Muchas tecnologías alternativas de las que quizá no haya oído hablar eran más rápidas que Ethernet cuando se introdujeron. Junto con ATM, esta lista incluye la FDDI (Interfaz de Datos Distribuidos por Fibra, del inglés *Fiber Distributed Data Interface*) y el Canal de fibra,[†] dos redes LAN ópticas basadas en anillos. Ambas eran incompatibles con Ethernet. Ninguna de ellas tuvo éxito. Eran demasiado complicadas, por lo cual se requerían chips complejos y precios altos. La lección que se debió de aprender aquí era KISS (Mantenlo simple, del inglés *Keep It Simple, Stupid*). Con el tiempo, Ethernet se emparejó con ellas en términos de velocidad, a menudo pidiendo prestada una parte de su tecnología; por ejemplo, la codificación 4B/5B de FDDI y la codificación 8B/10B del Canal de fibra. Así, no les quedaron ventajas y murieron en forma silenciosa o quedaron en roles especializados.

Tal parece que Ethernet seguirá expandiendo sus aplicaciones durante un poco más de tiempo. Gracias al estándar Ethernet de 10 gigabits, se liberó de las restricciones de distancia de CSMA/CD. Se ha realizado un esfuerzo considerable en la **Ethernet con grado de portadora** para que los proveedores de red puedan ofrecer servicios basados en Ethernet a sus clientes para las redes de área metropolitana y amplia (Fouli y Maler, 2009). Esta aplicación transmite las tramas Ethernet por largas distancias a través de fibra óptica y exige mejores características de administración para ayudar a los operadores a ofrecer servicios confiables de alta calidad. Las redes de muy alta velocidad también se están empezando a usar planos posteriores para conectar componentes en grandes enrutadores o servidores. Estos usos son adicionales al envío de tramas entre computadoras ubicadas en oficinas.

[†] En inglés se le llama *Fibre Channel* y no *Fiber Channel*, debido a que el editor del documento era británico.

4.4 REDES LAN INALÁMBRICAS

Las redes LAN inalámbricas son cada vez más populares; los hogares, oficinas, cafeterías, bibliotecas, aeropuertos y demás sitios públicos se están equipando con este tipo de redes para conectar computadoras, dispositivos PDA y teléfonos inteligentes (*smartphones*) a Internet. Las redes LAN inalámbricas también se pueden usar para permitir que dos o más computadoras que estén cerca unas de otras se comuniquen sin necesidad de usar Internet.

El principal estándar de LAN inalámbrica es 802.11. En la sección 1.5.3 vimos algunos de sus antecedentes. Ahora es tiempo de analizar la tecnología con detalle. En las siguientes secciones analizaremos la pila de protocolos, las técnicas de radiotransmisión de la capa física, el protocolo de subcapa MAC, la estructura de las tramas y los servicios ofrecidos. Si desea más información sobre el estándar 802.11, consulte a Gast (2005). Si quiere "escuchar" la verdad de primera mano, consulte el estándar publicado IEEE 802.11-2007.

4.4.1 La arquitectura de 802.11 y la pila de protocolos

Las redes 802.11 se pueden utilizar en dos modos. El modo más popular es conectar clientes, como laptops y teléfonos inteligentes, a otra red, como la intranet de una empresa o Internet. Este modo se muestra en la figura 4-23(a). En el modo de infraestructura, cada cliente se asocia con un **AP** (**Punto de Acceso**, del inglés *Access Point*) que a su vez está conectado a la otra red. El cliente envía y recibe sus paquetes a través del AP. Se pueden conectar varios puntos de acceso juntos, por lo general mediante una red alámbrica llamada **sistema de distribución**, para formar una red 802.11 extendida. En este caso, los clientes pueden enviar tramas a otros clientes a través de sus APs.

El otro modo, que se muestra en la figura 4-23(b), es una **red *ad hoc***. Este modo es una colección de computadoras que están asociadas de manera que puedan enviarse tramas directamente unas a otras. No hay punto de acceso. Como el acceso a Internet es la aplicación esencial para las redes inalámbricas, las redes ad hoc no son muy populares.

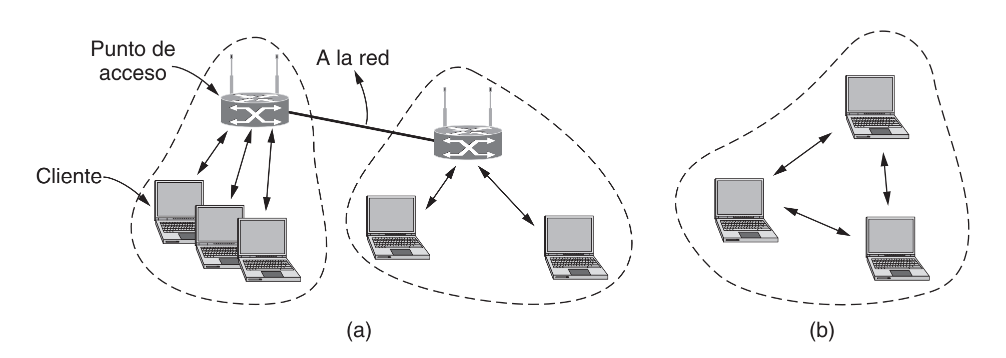


Figura 4-23. Arquitectura 802.11. (a) Modo de infraestructura. (b) Modo *ad hoc*.

Ahora analizaremos los protocolos. Todos los protocolos 802, incluyendo 802.11 y Ethernet, tienen ciertas similitudes en su estructura. En la figura 4-24 se muestra una vista parcial de la pila de protocolos del estándar 802.11. La pila es la misma para los clientes y APs. La capa física corresponde muy bien con la capa física OSI, pero la capa de enlace de datos en todos los protocolos 802 se divide en dos o más subcapas. En el estándar 802.11, la subcapa MAC (Control de Acceso al Medio) determina la forma en que se asigna el canal; es decir, a quién le toca transmitir a continuación. Arriba de dicha subcapa se encuentra

la subcapa LLC (Control de Enlace Lógico), cuya función es ocultar las diferencias entre las variantes 802 con el fin de que sean imperceptibles en lo que respecta a la capa de red. Ésta podría haber sido una responsabilidad considerable, pero en estos días la LLC es una capa de pegamento que identifica el protocolo (por ejemplo, IP) que se transporta dentro de una trama 802.11.

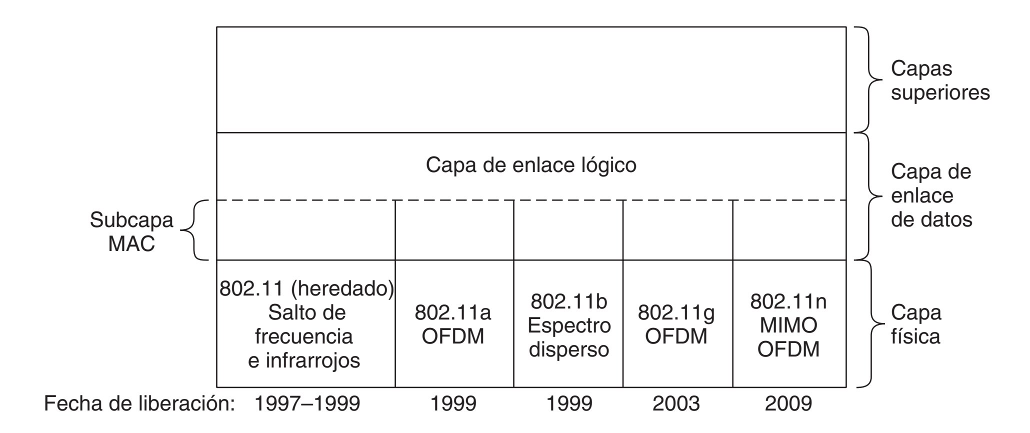


Figura 4-24. Parte de la pila de protocolos 802.11.

Desde su aparición por primera vez en 1997, se han agregado varias técnicas de transmisión a la capa física a medida que el 802.11 ha ido evolucionando. Dos de las técnicas iniciales, infrarrojos en la forma de los controles remotos de televisión y el salto de frecuencia en la banda de 2.4 GHz, están ahora extintos. La tercera técnica inicial, el espectro disperso de secuencia directa a 1 o 2 Mbps en la banda de 2.4 GHz, se extendió para operar a tasas de hasta 11 Mbps y se convirtió rápidamente en un éxito. Ahora se conoce como 802.11b.

Para dar a los amantes de las redes inalámbricas el tan deseado aumento de velocidad, se introdujeron en 1999 y 2003 nuevas técnicas de transmisión basadas en el esquema OFDM (Multiplexión por División de Frecuencia Ortogonal) que describimos en la sección 2.5.3. La primera se llama 802.11a y utiliza una banda de frecuencia distinta, 5 GHz. La segunda se quedó con la banda de 2.4 GHz y la compatibilidad. Se llama 802.11g. Ambas ofrecen tasas de transmisión de hasta 54 Mbps.

En octubre de 2009 finalizó la creación de unas técnicas de transmisión que utilizan varias antenas en forma simultánea en el transmisor y el receptor para aumentar la velocidad, conocidas como 802.11n. Con cuatro antenas y canales más amplios, el estándar 802.11 ahora define tasas de transmisión asombrosas de hasta 600 Mbps.

Ahora examinaremos con brevedad cada una de estas técnicas de transmisión. Sin embargo, sólo cubriremos las que están en uso y omitiremos los métodos de transmisión 802.11 heredados. Técnicamente, éstos pertenecen a la capa física y deberíamos haberlos examinado en el capítulo 2, pero como están tan estrechamente enlazados con las redes LAN en general y con la LAN 802.11 en particular, los veremos mejor aquí.

4.4.2 La capa física del estándar 802.11

Cada una de las cinco técnicas de transmisión hace que sea posible enviar una trama MAC por el aire, de una estación a otra. Sin embargo, difieren en la tecnología utilizada y en las velocidades alcanzables. Un estudio detallado de estas tecnologías está más allá del alcance de este libro, pero unas cuantas palabras sobre dichas tecnologías relacionarán las técnicas con el material que cubrimos en la sección 2.5 y proveerán a los lectores interesados algunos términos clave con los cuales podrán buscar más información en alguna otra parte.

Todas las técnicas 802.11 usan radios de corto alcance para transmitir señales en las bandas de frecuencias ISM de 2.4 GHz o de 5 GHz, las cuales se describen en la sección 2.3.3. Estas bandas poseen la ventaja de que no necesitan licencia y por ende, están libremente disponibles para cualquier transmisor que desee cumplir con ciertas restricciones, como una potencia radiada de al menos 1 W (aunque es más típico el valor de 50 mW para los radios LAN inalámbricos). Por desgracia, este hecho también lo conocen los fabricantes de abridores de puertas de cochera, teléfonos inalámbricos, hornos de microondas y numerosos dispositivos más, los cuales compiten con las computadoras portátiles por el mismo espectro. La banda de 2.4 GHz tiende a estar más saturada que la banda de 5 GHz, por lo que tal vez esta última sea mejor para ciertas aplicaciones, aun cuando tiene un alcance más corto debido a que la frecuencia es más alta.

Además, todos los métodos de transmisión definen múltiples tasas. La idea es que se puedan utilizar distintas tasas dependiendo de las condiciones actuales. Si la señal inalámbrica es débil, se puede usar una tasa baja. Si la señal es clara, se puede usar la tasa más alta. A este ajuste se le conoce como **adaptación de tasa**. Ya que las tasas varían por un factor de 10 o más, es importante una buena adaptación de tasa para un buen desempeño. Desde luego que, como no se necesita para la interoperabilidad, los estándares no dicen cómo se debe realizar la adaptación de tasa.

El primer método de transmisión que analizaremos es **802.11b**. Es un método de espectro disperso que soporta tasas de 1, 2, 5.5 y 11 Mbps, aunque en la práctica la tasa de operación es casi siempre de 11 Mbps. Es similar al sistema CDMA que examinamos en la sección 2.5, excepto que sólo hay un código de dispersión que comparten todos los usuarios. La dispersión se utiliza para satisfacer el requerimiento de la FCC que establece que se debe dispersar la potencia a través de la banda ISM. La secuencia de dispersión que utiliza el 802.11b es una **secuencia de Barker**. Tiene la propiedad de que su autocorrelación es baja, excepto cuando las secuencias están alineadas. Esta propiedad permite a un receptor bloquear el inicio de una transmisión. Para enviar a una tasa de 1 Mbps, se utiliza la secuencia de Barker con modulación BPSK para enviar 1 bit por 11 chips. Los chips se transmiten a una tasa de 11 Mchips/seg. Para enviar a 2 Mbps, se utiliza con modulación QPSK para enviar 2 bits por 11 chips. Las tasas más altas son distintas. Estas tasas usan una técnica llamada **CCK** (**Modulación por Código Complementario**, del inglés *Complementary Code Keying*) para construir códigos en vez de la secuencia de Barker. La tasa de 5.5 Mbps envía 4 bits en cada código de 8 chips, y la tasa de 11 Mbps envía 8 bits en cada código de 8 chips.

Ahora hablaremos sobre el **802.11a**, que soporta tasas de hasta 54 Mbps en la banda ISM de 5 GHz. Tal vez piense que el 802.11a debe ir antes que el 802.11b, pero en este caso no fue así. Aunque el grupo 802.11a se estableció primero, el estándar 802.11b se aprobó antes y su producto llegó al mercado mucho antes de los productos 802.11a, en parte debido a la dificultad de operar en la banda más alta de 5 GHz.

El método 802.11a se basa en **OFDM** (**Multiplexión por División de Frecuencia Ortogonal**, del inglés *Orthogonal Frequency Division Multiplexing*), ya que OFDM usa el espectro con eficiencia y resiste las degradaciones de las señales inalámbricas tales como multitrayectoria. Los bits se envían a través de 52 subportadoras en paralelo, 48 de las cuales llevan datos y 4 se usan para sincronización. Cada símbolo dura 4μs y envía 1, 2, 4 o 6 bits. Los bits están codificados para corrección de errores mediante un código convolucional primero, por lo que sólo 1/2, 2/3 o 3/4 partes de los bits no son redundantes. Con distintas combinaciones, el 802.11a puede operar a ocho tasas distintas, que varían desde 6 hasta 54 Mbps. Estas tasas son considerablemente más rápidas que las tasas del 802.11b, además de que hay menos interferencia en la banda de 5 GHz. Sin embargo, el 802.11b tiene un alcance aproximado de siete veces mayor que el del 802.11a, lo cual es más importante en muchas situaciones.

Incluso con el mayor alcance, los encargados del 802.11b no tenían intenciones de permitir que los presuntuosos del 802.11a ganaran el campeonato de velocidad. Por fortuna, en mayo de 2002 la FCC

retiró su vieja regla que exigía a todos los equipos de comunicaciones inalámbricas que operaban en las bandas ISM en Estados Unidos usar el espectro disperso, así que se puso a trabajar en el **802.11g**, que fue aprobado por el IEEE en 2003. Este estándar copia los métodos de modulación OFDM del 802.11a, pero opera en la banda ISM estrecha de 2.4 GHz junto con el 802.11b. Ofrece las mismas tasas de transmisión que el 802.11a (de 6 a 54 Mbps), además de compatibilidad con cualquier dispositivo 802.11b que se encuentre cerca. Todas estas distintas opciones pueden ser confusas para los clientes, por lo que es común que los productos soporten 802.11a/b/g en una sola NIC.

No conforme con detenerse ahí, el comité del IEEE empezó a trabajar en una capa física con una tasa de transferencia real alta, conocida como el estándar **802.11n**, el cual se ratificó en 2009. El objetivo del 802.11n era una tasa real de transferencia de por lo menos 100 Mbps después de eliminar todas las sobrecargas inalámbricas. Este objetivo exigía un aumento de por lo menos cuatro veces la velocidad en crudo. Para hacerlo realidad, el comité duplicó los canales de 20 MHz a 40 MHz y redujo las sobrecargas de entramado al permitir enviar todo un grupo de tramas a la vez. Pero es más notable el hecho de que el 802.11n usa hasta cuatro antenas para transmitir hasta cuatro flujos de información a la vez. Las señales de los flujos interfieren en el receptor, pero se pueden separar mediante técnicas de comunicaciones **MIMO** (**Múltiples Entrada Múltiples Salida**, del inglés *Multiple Input Multiple Output*). El uso de varias antenas ofrece un enorme aumento en la velocidad, o en su defecto un mejor alcance y confiabilidad. Al igual que OFDM, MIMO es una de esas astutas ideas de comunicaciones que está cambiando los diseños inalámbricos y que quizá vaya a ser muy popular en lo futuro. Para una breve introducción a las antenas múltiples en el 802.11, consulte a Halperin y colaboradores (2010).

4.4.3 El protocolo de la subcapa MAC del 802.11

Regresemos ahora de la tierra de la ingeniería eléctrica a la de las ciencias de la computación. El protocolo de la subcapa MAC para el estándar 802.11 es muy diferente al de Ethernet, debido a dos factores fundamentales para la comunicación inalámbrica.

Primero, las radios casi siempre son half-dúplex, lo cual significa que no pueden transmitir y escuchar ráfagas de ruido al mismo tiempo en una sola frecuencia. La señal recibida puede ser un millón de veces más débil que la señal transmitida, por lo que no se puede escuchar al mismo tiempo. Con Ethernet, una estación sólo espera hasta que el medio queda en silencio y después comienza a transmitir. Si no recibe una ráfaga de ruido mientras transmite los primeros 64 bytes, es muy probable que se haya entregado la trama correctamente. En los sistemas inalámbricos, este mecanismo de detección de colisiones no funciona.

En vez de ello, el 802.11 trata de evitar colisiones con un protocolo llamado **CSMA/CA** (**CSMA con Evitación de Colisiones**, del inglés *CSMA with Collision Avoidance*). En concepto, este protocolo es similar al CSMA/CD de Ethernet, con detección del canal antes de enviar y retroceso exponencial después de las colisiones. Sin embargo, una estación que desee enviar una trama empieza con un retroceso aleatorio (excepto en el caso en que no haya utilizado el canal recientemente y éste se encuentre inactivo). No espera una colisión. El número de ranuras para el retroceso se elije en el rango de 0 hasta, por decir, 15 en el caso de la capa física OFDM. La estación espera hasta que el canal está inactivo, para lo cual detecta que no hay señal durante un periodo corto (llamado DIFS, como veremos más adelante) y realiza un conteo descendente de las ranuras inactivas, haciendo pausa cuando se envían tramas. Envía su trama cuando el contador llega a 0. Si la trama logra pasar, el destino envía de inmediato una confirmación de recepción corta. La falta de una confirmación de recepción se interpreta como si hubiera ocurrido un error, sea una colisión o cualquier otra cosa. En este caso, el emisor duplica el periodo de retroceso e intenta de nuevo, continuando con el retroceso exponencial como en Ethernet, hasta que la trama se transmita con éxito o se llegue al número máximo de retransmisiones.

En la figura 4-25 se muestra una línea de tiempo de ejemplo. La estación *A* es la primera en enviar una trama. Mientras *A* envía, las estaciones *B* y *C* se preparan para enviar. Ven que el canal está ocupado y esperan a que esté inactivo. Poco después de que *A* recibe una confirmación de recepción, el canal queda inactivo. Sin embargo, en vez de enviar una trama de inmediato y colisionar, *B* y *C* realizan un retroceso. *C* elije un retroceso corto, por lo que envía primero. *B* detiene su conteo mientras detecta que *C* está usando el canal y lo reanuda después de que *C* recibe una confirmación de recepción. Poco después, *B* completa su retroceso y envía su trama.

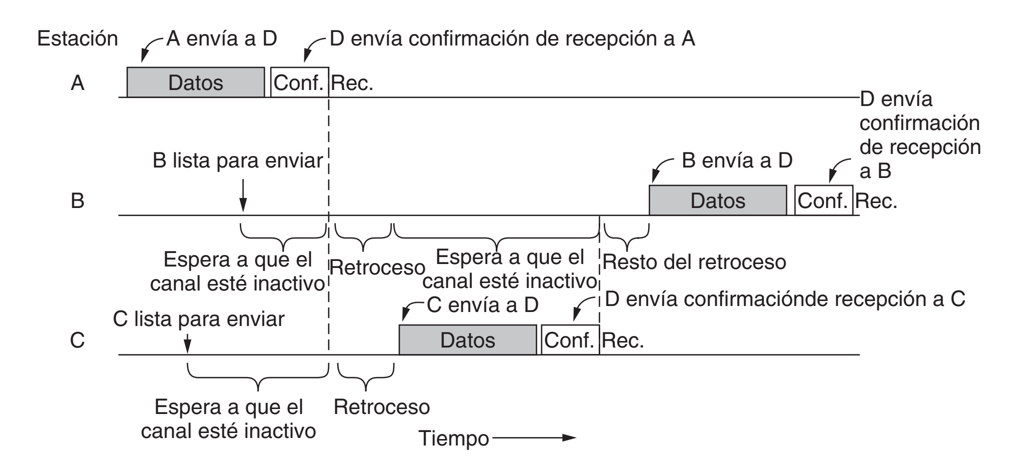


Figura 4-25. Envío de una trama mediante CSMA/CA.

En comparación con Ethernet, hay dos diferencias principales. Primero, empezar los retrocesos lo más pronto posible ayuda a evitar las colisiones. Esto vale la pena debido a que las colisiones son costosas, puesto que se transmite toda la trama incluso aunque ocurra una colisión. Segundo, se utilizan confirmaciones de recepción para inferir colisiones, ya que estas últimas no se pueden detectar.

Este modo de operación se llama **DCF** (**Función de Coordinación Distribuida**, del inglés *Distributed Coordination Function*), ya que cada estación actúa en forma independiente, sin ningún tipo de control central. El estándar también incluye un modo opcional de operación llamado **PCF** (**Función de Coordinación Puntual**, del inglés *Point Coordination Function*), en donde el punto de acceso controla toda la actividad en su celda, justo igual que una estación base celular. Sin embargo, PCF no se utiliza en la práctica debido a que por lo general no hay forma de evitar que las estaciones en otra red cercana transmitan tráfico conflictivo.

El segundo problema es que los rangos de transmisión de las distintas estaciones pueden ser diferentes. Con un cable, el sistema se diseña de tal forma que todas las estaciones se puedan escuchar entre sí. Con las complejidades de la propagación de RF, esta situación no es válida para las estaciones inalámbricas. En consecuencia, pueden surgir situaciones como el problema de la terminal oculta que vimos antes y que ilustramos de nuevo en la figura 4-26(a). Como no todas las estaciones están dentro del alcance de radio de todas las demás, las transmisiones que se realizan en una parte de una celda tal vez no se reciban en las demás partes de la misma celda. En este ejemplo, la estación *C* transmite a la estación *B*. Si *A* detecta el canal, no escuchará nada y concluirá de manera errónea que en ese momento puede empezar a transmitir a *B*. Esta decisión provoca una colisión.

La situación inversa es el problema de la terminal expuesta, que se ilustra en la figura 4-26(b). Aquí, *B* desea enviar a *C*, por lo que escucha en el canal. Cuando oye una transmisión, concluye erróneamente que no puede enviar a *C*, aun cuando *A* puede en ese momento estar transmitiendo a *D* (esto no se muestra). Esta decisión desperdicia una oportunidad de transmisión.

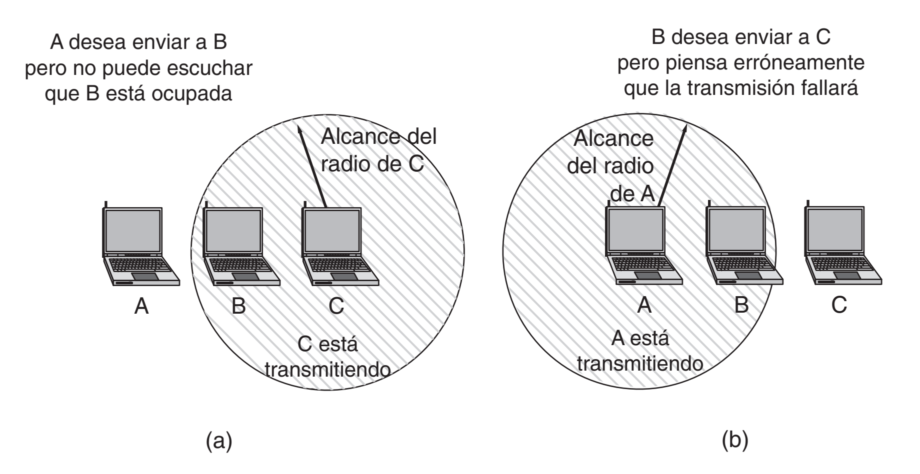


Figura 4-26. (a) El problema de la terminal oculta. (b) El problema de la terminal expuesta.

Para reducir las ambigüedades con respecto a qué estación va a transmitir, el 802.11 define la detección del canal como un proceso que consiste tanto de una detección física como de una detección virtual. En la detección física sólo se verifica el medio para ver si hay una señal válida. En la detección virtual, cada estación mantiene un registro lógico del momento en que se usa el canal rastreando el **NAV** (**Vector de Asignación de Red**, del inglés *Network Allocation Vector*). Cada trama lleva un campo NAV que indica cuánto tiempo tardará en completarse la secuencia a la que pertenece esta trama. Las estaciones que escuchen por casualidad esta trama saben que el canal estará ocupado durante el periodo indicado por el NAV, sin importar que puedan detectar o no una señal física. Por ejemplo, el NAV de una trama de datos indica el tiempo necesario para enviar una confirmación de recepción. Todas las estaciones que escuchen la trama de datos se retardarán durante el periodo de la confirmación de recepción, puedan o no escucharla.

Hay un mecanismo RTS/CTS opcional que usa el NAV para evitar que las terminales envíen tramas al mismo tiempo como terminales ocultas. Este mecanismo se muestra en la figura 4-27. En este ejemplo, *A* desea enviar a *B*. *C* es una estación dentro del alcance de *A* (y posiblemente dentro del alcance de *B*, pero eso no importa). *D* es una estación dentro del alcance de *B*, pero no dentro del alcance de *A*.

El protocolo empieza cuando *A* decide que desea enviar datos a *B*. *A* empieza por enviar una trama RTS a *B* para solicitar permiso de enviarle una trama. Si *B* recibe esta solicitud, responde con una trama CTS para indicar que el canal está libre para enviar. Al recibir la CTS, *A* envía su trama e inicia un temporizador ACK. Al recibir de forma correcta la trama de datos, *B* responde con una trama ACK para completar el intercambio. Si el temporizador ACK de *A* expira antes de que la trama ACK vuelva a ella, se considera como una colisión y se lleva a cabo todo el protocolo de nuevo, después de un retroceso.

Ahora consideremos este intercambio desde los puntos de vista de *C* y *D*. *C* está dentro del alcance de *A*, por lo que puede recibir la trama RTS. Si pasa esto, se da cuenta de que alguien pronto va a enviar datos. A partir de la información proporcionada en la solicitud RTS, C puede estimar cuánto tardará la secuencia, incluyendo la trama ACK final. Entonces, por el bien de todos desiste de transmitir cualquier cosa hasta que el intercambio esté completo. A continuación actualiza su registro del NAV para indicar que el canal está ocupado, como se muestra en la figura 4-27. *D* no escucha el RTS pero sí el CTS, por lo que también actualiza su NAV. Es necesario tener en cuenta que las señales NAV no se transmiten; sólo son recordatorios internos para mantenerse en silencio durante cierto periodo.

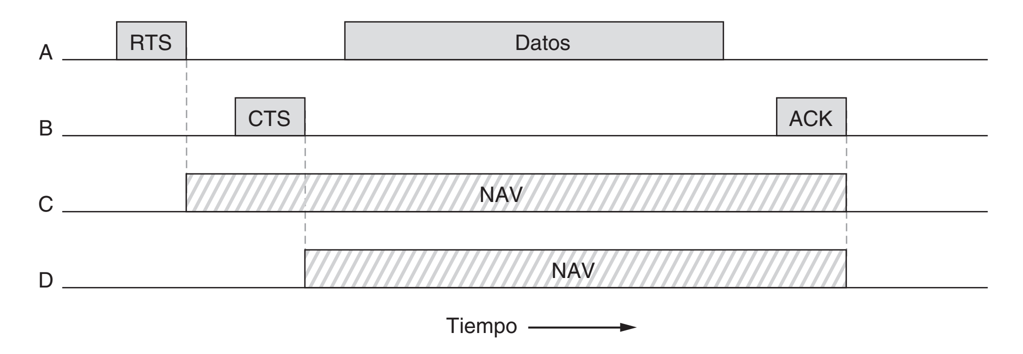


Figura 4-27. Detección de canal virtual mediante CSMA/CA.

Pero aunque el método RTS/CTS suena bien en teoría, es uno de esos diseños que ha demostrado ser de poco valor en la práctica. Se conocen varias razones por las que se utiliza pocas veces. No es útil para las tramas cortas (que se envían en vez de la trama RTS) ni para el AP (que todos pueden escuchar, por definición). Para otras situaciones, sólo reduce la operación. El método RTS/CTS en 802.11 es un poco distinto al del protocolo MACA que vimos en la sección 4.2, ya que todo el que escucha la trama RTS o CTS permanece en silencio para permitir que la trama ACK llegue a su destino sin que haya una colisión. Debido a esto, no es útil con las terminales expuestas como en el caso de MACA, sólo con las terminales ocultas. Lo más frecuente es que haya unas cuantas terminales ocultas y CSMA/CA les ayuda al reducir la velocidad de las estaciones que transmiten sin éxito, sin importar cuál sea la causa, para que sus transmisiones tengan más probabilidades de tener éxito.

CSMA/CA con detección física y virtual es el núcleo del protocolo 802.11. Sin embargo, existen otros mecanismos que se han desarrollado para trabajar con él. Cada uno de estos mecanismos fue impulsado por las necesidades de la operación real, por lo que los analizaremos de manera breve.

La primera necesidad que analizaremos es la confiabilidad. En contraste con las redes convencionales de cables, las redes inalámbricas son ruidosas y poco confiables, lo cual se debe en gran parte a la interferencia de otros tipos de dispositivos como los hornos de microondas, que también usan las bandas ISM sin licencia. El uso de confirmaciones de recepción y retransmisiones es de poca ayuda si es baja la probabilidad de lograr que una trama llegue a su destino es poca, en primer lugar.

La principal estrategia que se utiliza en este caso para incrementar las transmisiones exitosas es reducir la tasa de transmisión. Las tasas más bajas usan modulaciones más robustas que tienen mayor probabilidad de ser recibidas correctamente para una relación señal a ruido dada. Si se pierden demasiadas tramas, una estación puede reducir la tasa. Si se entregan tramas con pocas pérdidas, la estación puede algunas veces probar una tasa más alta para ver si es conveniente usarla.

Otra estrategia para mejorar la probabilidad de que la trama llegue sin daños es enviar tramas más cortas. Si la probabilidad de que haya un bit con error es p, la probabilidad de que toda una trama de n bits se reciba correctamente es $(1 - p)^n$. Por ejemplo, para $p = 10^{-4}$, la probabilidad de recibir correctamente una trama Ethernet completa (12 144 bits) es menor del 30%. Se perderá la mayoría de las tramas. Pero si la longitud de las tramas es de sólo un tercio (4 048 bits), dos tercios de ellas se recibirán correctamente. Ahora la mayoría de las tramas llegarán a su destino y se necesitarán menos retransmisiones.

Para implementar tramas más cortas es necesario reducir el tamaño máximo del mensaje que se aceptará de la capa de red. Como alternativa, el 802.11 permite dividir las tramas en piezas más pequeñas llamadas **fragmentos**, cada una con su propia suma de verificación. El tamaño del fragmento no es fijo según el estándar, sino un parámetro que el AP puede ajustar. Los fragmentos se enumeran en forma individual y su confirmación de recepción se realiza mediante un protocolo de parada y espera (es decir, el emisor no puede transmitir el fragmento $k + 1$ sino hasta que haya recibido la confirmación de recepción para el fragmento k). Una vez que se adquiere el canal, se envían los múltiples fragmentos como una ráfaga.

Van uno después del otro con una confirmación de recepción (y posiblemente retransmisiones) entre ellos, hasta que se haya enviado con éxito toda la trama o el tiempo de transmisión llegue al máximo permitido. El mecanismo NAV mantiene a las demás estaciones en silencio sólo hasta la siguiente confirmación de recepción, pero se utiliza otro mecanismo (que veremos a continuación) para permitir el envío de una ráfaga de fragmentos sin que otras estaciones envíen una trama a la mitad de la transmisión.

La segunda necesidad que estudiaremos es el ahorro de energía. La vida de las baterías siempre es un asunto importante para los dispositivos inalámbricos móviles. El estándar 802.11 pone atención a la cuestión de la administración de energía, de modo que los clientes no tengan que desperdiciarla cuando no tengan información qué enviar o recibir.

El mecanismo básico para ahorrar energía se basa en las **tramas baliza** (*beacon frames*). Las balizas son difusiones periódicas que realiza el AP (por ejemplo, cada 100 mseg). Las tramas anuncian la presencia del AP a los clientes y llevan los parámetros del sistema, como el identificador del AP, el tiempo, cuánto falta para la siguiente baliza y la configuración de seguridad.

Los clientes pueden establecer un bit de administración de energía en las tramas que envían al AP para indicarle que entrarán en el **modo de ahorro de energía**. En este modo, el cliente puede dormitar y el AP pondrá en el búfer el tráfico destinado a este cliente. Para verificar el tráfico entrante, el cliente se despierta durante cada baliza y verifica un mapa de tráfico que se envía como parte de ella. Este mapa indica al cliente si hay tráfico en el búfer. De ser así, el cliente envía un mensaje de sondeo al AP, quien a su vez le envía el tráfico que está en el búfer. Después el cliente puede regresar al modo suspendido hasta que se envíe la siguiente baliza.

En 2005 se agregó al estándar 802.11 otro mecanismo de ahorro de energía, conocido como **APSD** (**Entrega Automática con Ahorro de Energía**, del inglés *Automatic Power Save Delivery*). Con este nuevo mecanismo, el AP coloca las tramas en el búfer y las envía a un cliente justo después de que éste envía tramas al AP. Así, el cliente puede regresar al modo suspendido hasta que tenga más tráfico para enviar (y recibir). Este mecanismo funciona bien para las aplicaciones como VoIP que tienen tráfico frecuente en ambas direcciones. Por ejemplo, un teléfono inalámbrico VoIP podría usarlo para enviar y recibir tramas cada 20 mseg, algo mucho más frecuente que el intervalo de baliza de 100 mseg, mientras dormita entre cada transmisión.

La tercera y última necesidad que examinaremos es la calidad del servicio. Cuando el tráfico VoIP en el ejemplo anterior compite con el tráfico de igual a igual, el primero es el que sufre, ya que se retrasará debido a la contención con el tráfico de igual a igual que requiere un ancho de banda alto, aun cuando el ancho de banda de VoIP es bajo. Es probable que los retardos degraden las llamadas de voz. Para evitar esta degradación, sería conveniente dejar que el tráfico VoIP vaya adelante del tráfico de igual a igual, puesto que es de mayor prioridad.

El estándar IEEE 802.11 tiene un ingenioso mecanismo para proveer este tipo de calidad de servicio, el cual se introdujo como un conjunto de extensiones bajo el nombre 802.11e en 2005. Su función consiste en extender el CSMA/CA con intervalos cuidadosamente definidos entre las tramas. Después de enviar una trama, se requiere cierta cantidad de tiempo inactivo antes de que una estación pueda enviar otra para verificar si el canal ya no se está usando. El truco es definir distintos intervalos para los distintos tipos de tramas.

En la figura 4-28 se ilustran cinco intervalos. El intervalo entre las tramas de datos regulares se conoce como **DIFS** (**Espaciado Entre Tramas DCF**, del inglés *DCF InterFrame Spacing*). Cualquier estación puede intentar adquirir el canal para enviar una nueva trama después de que el medio haya estado inactivo durante un tiempo DIFS. Se aplican las reglas de contención usuales y tal vez se requiera el retroceso exponencial binario si ocurre una colisión. El intervalo más corto es **SIFS** (**Espaciado Corto Entre Tramas**, del inglés *Short InterFrame Spacing*) y se utiliza para permitir que las partes en un diálogo sencillo tengan la oportunidad de tomar el primer turno. Algunos ejemplos son: permitir que el receptor envíe una trama ACK, otras secuencias de tramas de control como RTS y CTS, o permitir que un emisor transmita una ráfaga de fragmentos. Enviar el siguiente fragmento después de esperar sólo un tiempo SIFS es lo que evita que otra estación irrumpa con una trama a mitad del intercambio.

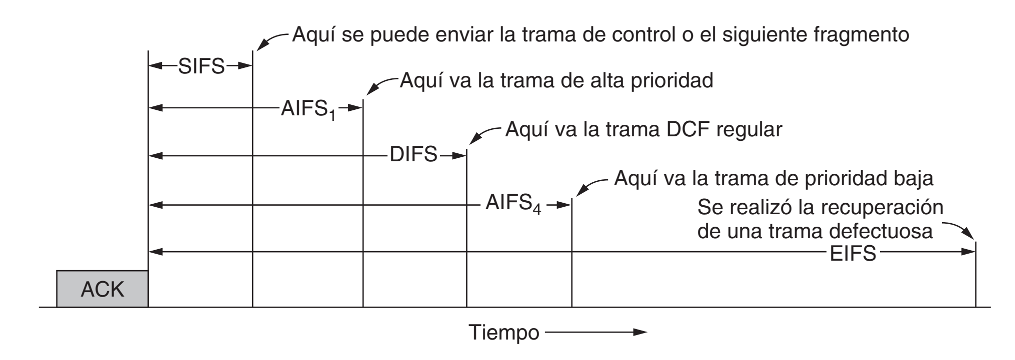


Figura 4-28. Espaciado entre tramas en el 802.11.

Los dos intervalos **AIFS** (**Espaciado Entre Tramas de Arbitraje**, del inglés *Arbitration InterFrame Space*) muestran ejemplos de dos niveles de prioridad distintos. El intervalo corto, $AIFS_1$, es más pequeño que el intervalo DIFS pero más largo que SIFS. El AP lo puede usar para transportar voz u otro tipo de tráfico de alta prioridad al inicio de la línea. El AP esperará un intervalo más corto antes de enviar el tráfico de voz, y por ende lo enviará antes del tráfico regular. El intervalo largo, $AIFS_4$, es más largo que DIFS. Se utiliza para el tráfico de fondo que se puede aplazar hasta después del tráfico regular. El AP esperará un intervalo más largo antes de enviar este tráfico, para dar al tráfico regular la oportunidad de transmitir primero. El mecanismo completo de calidad del servicio define cuatro distintos niveles de prioridad que tienen diferentes parámetros de retroceso, así como diferentes parámetros de inactividad.

El último intervalo, **EIFS** (**Espaciado Entre Tramas Extendido**, del inglés *Extended InterFrame Spacing*), lo utiliza sólo una estación que acaba de recibir una trama defectuosa o desconocida, para reportar el problema. La idea es que, como el receptor tal vez no tenga idea de lo que está sucediendo, debería esperar un poco para evitar interferir con un diálogo existente entre dos estaciones.

Una parte adicional de las extensiones de calidad del servicio es la noción de una **TXOP**, u **oportunidad de transmisión**. El mecanismo original CSMA/CA permitía que las estaciones enviaran una trama a la vez. Este diseño estaba bien hasta que aumentó el rango de las tasas de transmisión. En el 802.11a/g, una estación podría enviar a 6 Mbps y otra estación podría enviar a 54 Mbps. Las dos tienen oportunidad de enviar una trama, pero la estación de 6 Mbps se tarda nueve veces más (ignorando las sobrecargas fijas) que la estación de 54 Mbps en enviar su trama. Esta disparidad tiene el desafortunado efecto secundario de reducir la velocidad de un emisor rápido que compite con un emisor lento, a una tasa aproximada a la del emisor lento. Por ejemplo (e ignorando de nuevo las sobrecargas fijas), cuando los emisores de 6 Mbps y 54 Mbps transmiten cada uno por separado obtienen sus propias tasas, pero cuando envían juntos ambos obtienen 5.4 Mbps en promedio. Es un duro castigo para el emisor rápido. Este problema se conoce como la **anomalía de tasa** (Heusse y colaboradores, 2003).

Con las oportunidades de transmisión, cada estación recibe una cantidad equivalente de tiempo aire, no un número equivalente de tramas. Las estaciones que envían a una tasa más alta durante su tiempo aire obtendrán una velocidad de transmisión real más alta. En nuestro ejemplo, cuando los emisores de 6 Mbps y de 54 Mbps envíen juntos, obtendrán ahora 3 Mbps y 27 Mbps, respectivamente.

4.4.4 La estructura de trama 802.11

El estándar 802.11 define tres clases diferentes de tramas en el aire: de datos, de control y de administración. Cada una tiene un encabezado con una variedad de campos que se utilizan dentro de la subcapa

MAC. Además, hay algunos encabezados utilizados por la capa física, pero como éstos tienen que ver en su mayor parte con las técnicas de modulación utilizadas, no los trataremos aquí.

Analizaremos como ejemplo el formato de la trama de datos, que se muestra en la figura 4-29. Primero está el campo de *Control de trama*, que consta de 11 subcampos. El primero es la *Versión de protocolo*, que se establece como 00. Está ahí para que las futuras versiones del protocolo 802.11 funcionen al mismo tiempo en la misma celda. Después están los campos de *Tipo* (de datos, de control o de administración) y de *Subtipo* (por ejemplo, RTS o CTS). Para una trama de datos regular (sin calidad de servicio), se establecen en 10 y 0000 en binario. Los bits *Para DS* y *De DS* se establecen para indicar que la trama va hacia o viene de la red conectada a los APS, a la cual se le conoce como sistema de distribución. El bit *Más fragmentos* indica que siguen más fragmentos. El bit *Retransmitir* marca una retransmisión de una trama que se envió antes. El bit de *Administración de energía* indica que el emisor va a entrar al modo de ahorro de energía. El bit *Más datos* indica que el emisor tiene tramas adicionales para el receptor. El bit *Trama protegida* indica que el cuerpo de la trama se cifró por seguridad. En la siguiente sección hablaremos un poco sobre la seguridad. Por último, el bit de *Orden* indica al receptor que la capa superior espera que la secuencia de tramas llegue de modo riguroso en orden.

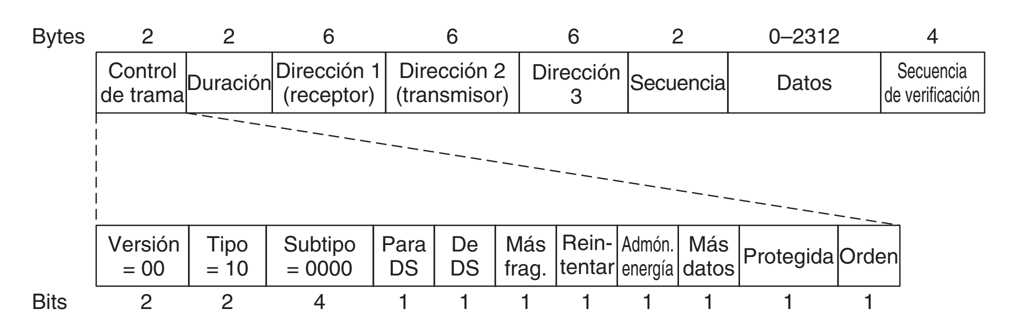


Figura 4-29. Formato de la trama de datos 802.11.

El segundo campo de la trama de datos, el campo *Duración*, indica cuánto tiempo ocuparán el canal la longitud de la trama y su confirmación de recepción, lo cual se mide en microsegundos. Está presente en todos los tipos de tramas, incluyendo las tramas de control, y es lo que utilizan las estaciones para administrar el mecanismo NAV.

Después siguen las direcciones. Las tramas de datos que se envían hacia o se reciben de un AP tienen tres direcciones, todas en formato estándar de IEEE 802. La primera dirección es el receptor, y la segunda dirección es el transmisor. Sin duda se necesitan pero, ¿para qué es la tercera dirección? Recuerde que el AP sólo es un punto de relevo para las tramas, a medida que viajan entre un cliente y otro punto en la red, tal vez un cliente distante o un portal de Internet. La tercera dirección provee este punto final distante.

El campo *Secuencia* numera las tramas de manera que se puedan detectar tramas duplicadas. De los 16 bits disponibles, 4 identifican el fragmento y 12 transportan un número que avanza con cada nueva transmisión. El campo *Datos* contiene la carga útil, hasta 2312 bytes. Los primeros bytes de esta carga útil están en un formato conocido como **LLC** (**Control de Enlace Lógico**, del inglés *Logical Link Control*). Esta capa es la unión que identifica al protocolo de capa superior (por ejemplo, IP) al que se deben pasar las cargas útiles. Por último tenemos la *Secuencia de verificación de tramas*, que viene siendo la misma CRC de 32 bits que vimos en la sección 3.2.2 y en otras partes.

Las tramas de administración tienen el mismo formato que las tramas de datos, además de un formato para la parte de los datos que varía con el subtipo (por ejemplo, los parámetros en las tramas de baliza).

Las tramas de control son cortas. Al igual que todas las tramas, tienen los campos *Control de trama*, *Duración* y *Secuencia de verificación de trama*. Sin embargo, ellas pueden tener sólo una dirección y ninguna porción de datos. La mayoría de la información clave se transmite mediante el campo *Subtipo* (por ejemplo, ACK, RTS y CTS).

4.4.5 Servicios

El estándar 802.11 define los servicios que los clientes, los puntos de acceso y la red que los conecta deben proveer para poder ser una LAN inalámbrica que se apegue a dicho estándar. Estos servicios se dividen en varios grupos.

El servicio de **asociación** lo utilizan las estaciones móviles para conectarse ellas mismas a los AP. Por lo general, se utiliza después de que una estación se mueve dentro del alcance de radio del AP. Al llegar, la estación conoce la identidad y las capacidades del AP, ya sea mediante tramas baliza o preguntando directamente al AP. Entre las capacidades se incluyen las tasas de datos soportadas, los arreglos de seguridad, las capacidades de ahorro de energía, el soporte de la calidad del servicio, etcétera. La estación envía una solicitud para asociarse con el AP. Éste puede aceptar o rechazar dicha solicitud.

La **reasociación** permite que una estación cambie su AP preferido. Esta herramienta es útil para las estaciones móviles que cambian de un AP a otro en la misma LAN 802.11 extendida, como un traspaso (*handover*) en la red celular. Si se utiliza en forma correcta, no se perderán datos como consecuencia del traspaso (pero al igual que Ethernet, el 802.11 es sólo un servicio de mejor esfuerzo). También es posible que la estación o el AP se **desasocien**, con lo que se rompería su relación. Una estación debe usar este servicio antes de desconectarse o salir de la red. El AP lo puede usar antes de desconectarse por cuestión de mantenimiento.

Las estaciones también se deben **autentificar** antes de poder enviar tramas por medio del AP, pero la autentificación se maneja en formas distintas dependiendo del esquema de seguridad elegido. Si la red 802.11 está "abierta", cualquiera puede usarla. En caso contrario, se requieren credenciales para autentificarse. El esquema recomendado, conocido como **WPA2** (**Acceso Protegido WiFi 2**, del inglés *WiFi Protected Access 2*), implementa la seguridad según lo definido en el estándar 802.11i (el WPA simple es un esquema interno que implementa un subconjunto del 802.11i. Lo omitiremos y pasaremos directo al esquema completo). En el WPA2, el AP se puede comunicar con un servidor de autentificación que tenga una base de datos con nombres de usuario y contraseñas para determinar si la estación puede acceder a la red. Como alternativa se puede configurar una clave precompartida, lo cual es un nombre elegante para una contraseña de red. Se intercambian varias tramas entre la estación y el AP con un reto y respuesta que permite a la estación demostrar que tiene las credenciales apropiadas. Este intercambio ocurre después de la asociación.

El esquema que se utilizaba antes de WPA se llama **WEP** (**Privacidad Equivalente a cableado**, del inglés *Wired Equivalent Privacy*). En este esquema, la autentificación con una clave precompartida ocurre antes de la asociación. Sin embargo, no se recomienda su uso debido a fallas de diseño que comprometen fácilmente su seguridad. La primera demostración práctica de que WEP tenía defectos se dio cuando Adam Stubblefield era un interno de verano en AT&T (Stubblefield y colaboradores, 2002). Él pudo codificar y probar un ataque en una semana, aunque la mayor parte de ese tiempo la invirtió en obtener permiso de la gerencia para comprar las tarjetas WiFi necesarias en los experimentos. Ahora hay software disponible libremente para quebrantar contraseñas WEP.

Una vez que las tramas llegan al AP, el servicio de **distribución** determina cómo encaminarlas. Si el destino es local para el AP, las tramas se pueden enviar en forma directa por el aire. En caso contrario, habrá que reenviarlas por la red alámbrica. El servicio de **integración** maneja cualquier traducción necesaria para enviar una trama fuera de la LAN 802.11, o para que llegue desde el exterior de la LAN 802.11. Aquí el caso común es conectar la LAN inalámbrica a Internet.

Y como lo esencial aquí es la transmisión de datos, es lógico que la red 802.11 provea el servicio de **entrega de datos**. Este servicio permite a las estaciones transmitir y recibir datos mediante el uso de los protocolos que describimos antes en este capítulo. Como el estándar 802.11 está modelado en base a Ethernet y no se garantiza que la transmisión por Ethernet sea 100% confiable, tampoco se garantiza que la transmisión por 802.11 sea confiable. Las capas superiores deben lidiar con la detección y corrección de errores.

La señal inalámbrica es de difusión. Para mantener confidencial la información que se envía por una LAN inalámbrica, hay que cifrarla. Para lograr esto se utiliza un servicio de **privacidad** que administra los detalles del cifrado y el descifrado. El algoritmo de cifrado para WPA2 se basa en el estándar **AES** (**Estándar de Cifrado Avanzado**, del inglés *Advanced Encryption Standard*) del gobierno de Estados Unidos, aprobado en 2002. Las claves que se utilizan para el cifrado se determinan durante el procedimiento de autentificación.

Para manejar tráfico con distintas prioridades, existe un servicio llamado **programación de tráfico QOS**, el cual utiliza los protocolos que describimos antes para dar al tráfico de voz y de video un tratamiento preferencial, en comparación con el tráfico del mejor esfuerzo y de fondo. Hay un servicio complementario que también provee la sincronización de los temporizadores de las capas superiores. Esto permite a las estaciones coordinar sus acciones, que pueden ser útiles para procesar los medios.

Por último hay dos servicios que ayudan a las estaciones a administrar la forma en que utilizan el espectro. El servicio de **control de potencia de transmisión** brinda a las estaciones la información que necesitan para cumplir con los límites regulatorios sobre la potencia de transmisión, que varían de una región a otra. El servicio de **selección de frecuencia dinámica** ofrece a las estaciones la información que necesitan para evitar transmitir en frecuencias de la banda de 5 GHz que se utilicen para radar en las proximidades.

Con estos servicios, el estándar 802.11 provee un extenso conjunto de funcionalidad para conectar los clientes móviles circundantes a Internet. Ha sido un enorme éxito, además de que el estándar se ha enmendado en repetidas ocasiones para agregar más funcionalidad. Para tener una perspectiva sobre lo que ha hecho el estándar y hacia donde se encamina, consulte a Hiertz y colaboradores (2010).

4.5 BANDA ANCHA INALÁMBRICA

Hemos estado en el interior demasiado tiempo. Vayamos al exterior, en donde hay muchas cosas interesantes sobre redes en lo que se conoce como la "última milla". Con la desregulación de los sistemas telefónicos en muchos países, ahora a los competidores de las compañías telefónicas arraigadas con frecuencia se les permite ofrecer servicios de voz local e Internet de alta velocidad. Sin duda hay mucha demanda. El problema es que el tendido de fibra óptica o cable coaxial a millones de casas y oficinas es muy costoso. ¿Qué puede hacer un competidor?

La respuesta es la banda ancha inalámbrica. Erigir una enorme antena en una colina en las afueras de la ciudad es más fácil y barato que cavar zanjas y ensartar cables. Por lo tanto, las compañías han empezado a experimentar y a ofrecer servicios de comunicación inalámbrica de megabits para voz, Internet, películas bajo demanda, etcétera.

Para estimular el mercado, el IEEE formó un grupo para estandarizar una red de área metropolitana inalámbrica de banda ancha. El siguiente número disponible en el espacio de numeración 802 era **802.16**, por lo que el estándar obtuvo este número. A esta tecnología se le conoce de manera informal como **WiMAX** (**Interoperabilidad Mundial para Acceso de Microondas**, del inglés *Worldwide Interoperability for Microwave Access*). Utilizaremos los términos 802.16 y WiMAX sin distinción.

El primer estándar 802.16 se aprobó en diciembre de 2001. Las primeras versiones proporcionaban un lazo local inalámbrico entre puntos fijos con una línea de visión entre sí. Pronto cambió este diseño

para hacer de WiMAX una alternativa más competitiva para el cable y DSL en cuanto al acceso a Internet. Para enero de 2003 se modificó el estándar 802.16 para que soportara enlaces que no estuvieran en la línea de visión, mediante el uso de tecnología OFDM a frecuencias entre 2 GHz y 10 GHz. Este cambio facilitó de manera considerable la implementación, aunque las estaciones seguían siendo fijas. El surgimiento de las redes celulares 3G representaba una amenaza, ya que prometía tasas de datos altas *y* movilidad. En respuesta a esto, el 802.16 se mejoró de nuevo para permitir movilidad con velocidades vehiculares en diciembre de 2005. El acceso a Internet móvil de banda ancha es el objetivo del estándar actual, IEEE 802.16-2009.

Al igual que los otros 802 estándares, el 802.16 estaba muy influenciado por el modelo OSI, incluyendo las (sub)capas, la terminología, las primitivas de servicio y otras características más. Por desgracia, también al igual que OSI, es bastante complicado. De hecho, se creó el **foro WiMAX** para definir subconjuntos interoperables del estándar para ofrecimientos comerciales. En las siguientes secciones veremos una breve descripción de algunos de los puntos sobresalientes de las formas comunes de la interfaz aérea 802.16, aunque este análisis dista mucho de estar completo y omite muchos detalles. Para obtener información adicional sobre WiMAX y el sistema inalámbrico de banda ancha en general, consulte a Andrews y colaboradores (2007).

4.5.1 Comparación del estándar 802.16 con 802.11 y 3G

En este momento tal vez piense: ¿por qué diseñar un nuevo estándar? ¿Por qué no sólo usar 802.11 o 3G? De hecho, WiMAX combina aspectos tanto de 802.11 como de 3G, lo cual lo convierte en algo más parecido a una tecnología 4G.

Al igual que 802.11, WiMAX trata exclusivamente sobre cómo conectar dispositivos en forma inalámbrica a Internet con velocidades de megabits/seg, en vez de usar cable o DSL. Los dispositivos pueden ser móviles, o por lo menos portátiles. WiMAX no empezó agregando datos de tasas bajas por el lado de las redes celulares de voz; el 802.16 se diseñó para transportar paquetes IP por el aire y conectarse a una red alámbrica basada en IP con relativa facilidad. Los paquetes pueden transportar tráfico de igual a igual, llamadas VoIP o medios de flujo continuo (*streaming*) para soportar una variedad de aplicaciones. Además y al igual que el 802.11, se basa en la tecnología OFDM para asegurar un buen rendimiento a pesar de las degradaciones de la señal inalámbrica como el desvanecimiento multitrayectoria, y también se basa en la tecnología MIMO para obtener altos niveles de tasas reales de transferencia.

Sin embargo, WiMAX es más parecido al estándar 3G (y por ende, distinto al 802.11) en varios aspectos clave. El problema técnico clave es lograr una capacidad alta mediante el uso eficiente del espectro, de modo que una gran cantidad de suscriptores en un área de cobertura puedan obtener altas tasas reales de transferencia. Las distancias típicas son por lo menos 10 veces más largas que para una red 802.11. En consecuencia, las estaciones base WiMAX son más poderosas que los puntos de acceso (AP) 802.11. Para manejar señales más débiles a través de distancias mayores, la estación base usa más potencia y mejores antenas, además de que realiza más labor de procesamiento para manejar los errores. Para maximizar la tasa real de transferencia, la estación base programa las transmisiones con cuidado para cada suscriptor específico; el uso del espectro no se deja al azar con CSMA/CA, ya que se puede desperdiciar la capacidad por las colisiones.

El espectro bajo licencia es el caso esperado para WiMAX, que por lo general está alrededor de los 2.5 GHz en Estados Unidos. Todo el sistema está mucho más optimizado que el 802.11. Esta complejidad vale la pena, tomando en cuenta la gran cantidad de dinero involucrada para el espectro bajo licencia. A diferencia del 802.11, el resultado es un servicio administrado y confiable, con un buen soporte para la calidad del servicio.

Con todas estas características, el 802.16 se asemeja más a las redes celulares 4G que ahora se están estandarizando bajo el nombre **LTE** (**Evolución a Largo Plazo**, del inglés *Long Term Evolution*). Mien-

tras que las redes celulares 3G se basan en CDMA y soportan voz y datos, las redes celulares 4G se basarán en OFDM con MIMO y estarán orientadas a los datos, dejando la voz como una simple aplicación. Parece que WiMAX y 4G van en una trayectoria de colisión en términos de tecnología y aplicaciones. Tal vez esta convergencia no sea sorpresa, dado que Internet es la aplicación revolucionaria y OFDM junto con MIMO son las tecnologías más conocidas para usar el espectro con eficiencia.

4.5.2 La arquitectura de 802.16 y la pila de protocolos

En la figura 4-30 se muestra la arquitectura del estándar 802.16. Las estaciones base se conectan de manera directa a la red troncal del proveedor, que a su vez se conecta a Internet. Las estaciones base se comunican con estaciones a través de la interfaz aérea inalámbrica. Existen dos tipos de estaciones. Las estaciones suscriptoras permanecen en una ubicación fija; por ejemplo, el acceso a Internet de banda ancha para los hogares. Las estaciones móviles pueden recibir servicio mientras se desplazan; por ejemplo, un auto equipado con WiMAX.

La pila de protocolos del estándar 802.16 que se utiliza a través de la interfaz de aire se muestra en la figura 4-31. La estructura general es similar a la de las otras redes 802, sólo que con más subcapas. La capa inferior se encarga de la transmisión; aquí sólo hemos mostrado los ofrecimientos populares de 802.16, WiMAX fijo y móvil. Hay una capa física distinta para cada ofrecimiento. Ambas capas operan en el espectro bajo licencia debajo de los 11 GHz y usan OFDM, pero en distintas formas.

Arriba de la capa física, la capa de enlace de datos consta de tres subcapas. La subcapa inferior se encarga de la privacidad y la seguridad, lo cual es mucho más crucial para las redes exteriores públicas que para las redes interiores privadas. Se encarga del cifrado, el descifrado y la administración de las claves.

A continuación tenemos la parte común de la subcapa MAC. En esta parte es en donde se encuentran los protocolos principales, como el de administración del canal. El modelo aquí es que la estación base controla el sistema por completo. Puede programar los canales del enlace descendente (es decir, de base a suscriptor) de una manera muy eficiente, además de que también desempeña un importante papel en la administración de los canales del enlace ascendente (es decir, de suscriptor a base). Una característica inusual de esta subcapa MAC es que, a diferencia de los otros protocolos 802, es completamente orientada a conexión, para poder proveer garantías de calidad del servicio para la comunicación de telefonía y multimedia.

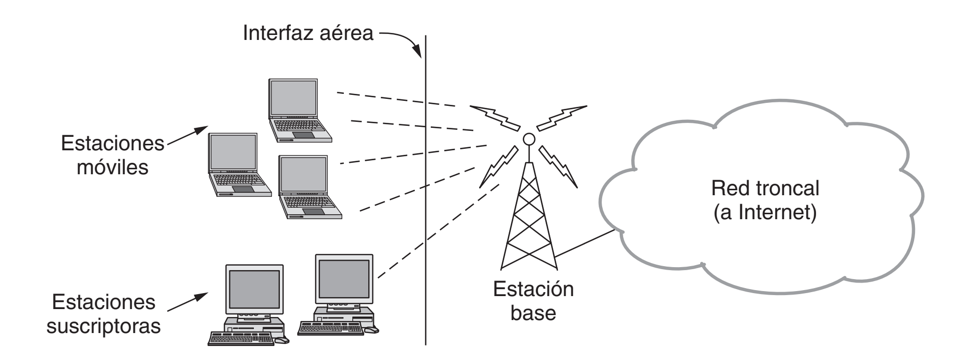


Figura 4-30. La arquitectura del estándar 802.16.

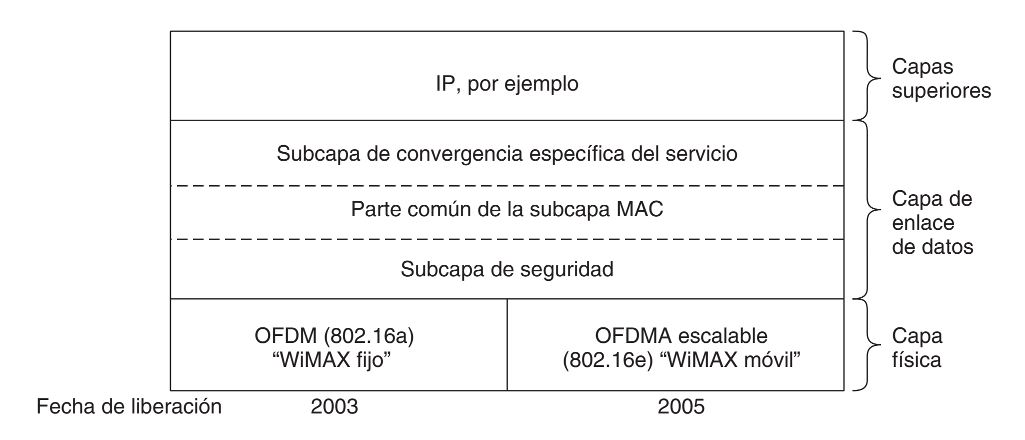


Figura 4-31. La pila de protocolos de 802.16.

La subcapa de convergencia específica del servicio toma el lugar de la subcapa de enlace lógico en los otros protocolos 802. Su función es proveer una interfaz para la capa de red. Las distintas capas de convergencia están definidas para integrarse sin problemas con las distintas capas superiores. La elección importante es IP, aunque el estándar también define asignaciones para protocolos como Ethernet y ATM. Como IP es un protocolo sin conexión y la subcapa MAC de 802.16 es orientada a conexión, esta capa debe realizar las asignaciones entre direcciones y conexiones.

4.5.3 La capa física del estándar 802.16

La mayoría de las implementaciones de WiMAX usan el espectro bajo licencia alrededor de los 3.5 GHz o los 2.5 GHz. Al igual que con 3G, encontrar espectro disponible es un problema clave. Para facilitar las cosas, el estándar 802.16 está diseñado con la flexibilidad en mente. Permite la operación de 2 GHz a 11 GHz. Se soportan canales de distintos tamaños; por ejemplo, 3.5 MHz para WiMAX fijo y de 1.25 MHz a 20 MHz para WiMAX móvil.

Las transmisiones se envían a través de estos canales mediante OFDM, la técnica que describimos en la sección 2.5.3. En comparación con el 802.11, el diseño OFDM del 802.16 está optimizado para obtener el máximo provecho del espectro bajo licencia y de las transmisiones de área amplia. El canal está dividido en más subportadoras con una mayor duración de los símbolos para tolerar niveles más altos de degradación en la señal inalámbrica; los parámetros de WiMAX son cerca de 20 veces más grandes que los parámetros comparables del estándar 802.11. Por ejemplo, en WiMAX móvil hay 512 subportadoras para un canal de 5 MHz y el tiempo para enviar un símbolo en cada subportadora es de alrededor de 100 μseg.

Los símbolos en cada subportadora se envían con los esquemas de modulación QPSK, QAM-16 o QAM-64 que describimos en la sección 2.5.3. Cuando la estación móvil o suscriptora está cerca de la estación base y la señal recibida tiene una relación señal a ruido (SNR) alta, se puede usar QAM-16 para enviar 6 bits por símbolo. Para llegar a estaciones distantes con una SNR baja, se puede usar QPSK para transmitir 2 bits por símbolo. Los datos se codifican primero para la corrección de errores mediante la codificación convolucional (o mejores esquemas) que describimos en la sección 3.2.1. Esta codificación es común en canales ruidosos para tolerar algunos errores de bits sin necesidad de enviar retransmisiones. De hecho, los métodos de modulación y codificación deben serle familiares para estos momentos, ya que se usan en muchas de las redes que hemos estudiado, incluyendo la red 802.11 de cable y DSL. El resultado neto es que una estación base puede soportar hasta 12.6 Mbps de tráfico descendente y 6.2 Mbps de tráfico ascendente por cada canal de 5 MHz y un par de antenas.

Algo que no les gustó a los diseñadores del estándar 802.16 era cierto aspecto de la forma en que funcionaban GSM y DAMPS. Ambos sistemas usan las mismas bandas de frecuencia para el tráfico ascendente y descendente. Es decir, suponen de manera implícita que hay la misma cantidad de tráfico tanto ascendente como descendente. Para la voz, el tráfico es simétrico en su mayor parte, pero para el acceso a Internet (y sin duda la navegación web) hay con frecuencia más tráfico descendente que ascendente. La proporción frecuente es de 2:1, 3:1 o más:1.

Así, los diseñadores eligieron un esquema flexible para dividir el canal entre varias estaciones, al cual se le conoce como **OFDMA** (**Acceso Múltiple por División de Frecuencia Ortogonal**, del inglés *Orthogonal Frequency Division Multiple Access*). Con el OFDMA se pueden asignar distintos conjuntos de subportadoras a distintas estaciones, de manera que más de una estación pueda enviar o recibir a la vez. Si fuera el estándar 802.11, una estación utilizaría todas las subportadoras para enviar en un momento dado. La flexibilidad adicional en cuanto a la forma en que se asigna el ancho de banda puede aumentar el desempeño, ya que una subportadora dada puede desvanecerse en un receptor debido a los efectos multitrayectoria, pero puede ser clara en otro receptor. Se pueden asignar subportadoras a las estaciones que puedan utilizarlas de la mejor forma.

Además de tener tráfico asimétrico, las estaciones comúnmente alternan entre el envío y la recepción. Este método se conoce como **TDD** (**Duplexión de División de Tiempo**, del inglés *Time Division Duplex*). El método alternativo, en donde una estación envía y recibe al mismo tiempo (en distintas frecuencias de subportadora), se llama **FDD** (**Duplexión de División de Frecuencia**, del inglés *Frequency Division Duplex*). WiMAX permite ambos métodos, pero se prefiere el TDD debido a que es más fácil de implementar y más flexible.

La figura 4-32 muestra un ejemplo de la estructura de trama que se repite con el tiempo. Empieza con un preámbulo para sincronizar todas las estaciones, seguido de transmisiones descendentes que provienen de la estación base. Primero, la estación base envía mapas que indican a todas las estaciones cómo se asignan las subportadoras descendentes y ascendentes a través de la trama. La estación base controla los mapas, por lo que puede asignar distintas cantidades de ancho de banda a las estaciones de trama en trama, dependiendo de las necesidades de cada estación.

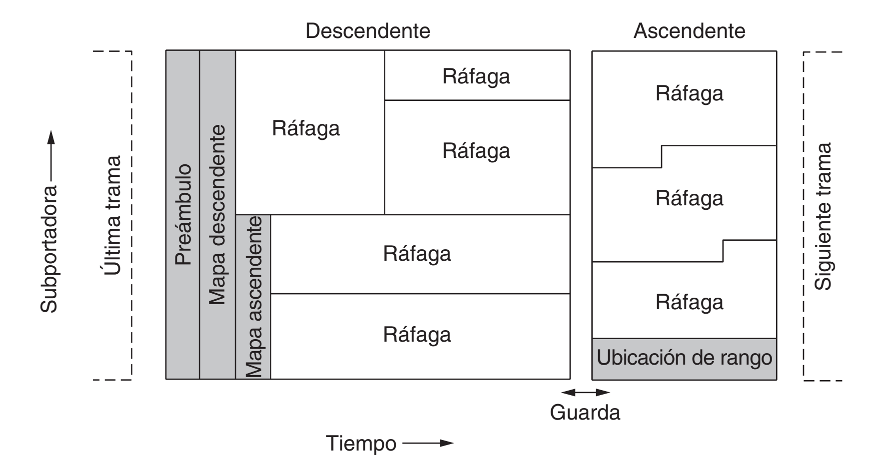


Figura 4-32. Estructura de trama para OFDMA con duplexión por división de tiempo.

A continuación, la estación base envía ráfagas de tráfico a distintas estaciones suscriptoras y móviles en las subportadoras, en los tiempos que se indican en el mapa. Las transmisiones descendentes terminan con un tiempo de guarda para que las estaciones cambien de recibir a transmitir. Por último, las estaciones

suscriptoras y móviles envían sus ráfagas de tráfico a la estación base en las posiciones ascendentes que se reservaron para ellas en el mapa. Una de estas ráfagas ascendentes se reserva para la **ubicación de rango** (*ranging*), que viene siendo el proceso a través del cual las nuevas estaciones ajustan su sincronización y solicitan el ancho de banda inicial para conectarse a la estación base. Como no se establece ninguna conexión en esta etapa, las nuevas estaciones sólo transmiten y esperan que no haya una colisión.

4.5.4 Protocolo de la subcapa MAC del estándar 802.16

Como vimos en la figura 4-31, la capa de enlace de datos se divide en tres subcapas. Puesto que estudiaremos la criptografía hasta el capítulo 8, es difícil explicar ahora cómo funciona la subcapa de seguridad. Basta decir que el cifrado se utiliza para mantener en secreto todos los datos transmitidos. Sólo se cifran las cargas útiles de las tramas; los encabezados no se cifran. Esta propiedad significa que un intruso puede ver quién está hablándole a quién, pero no puede saber qué se están diciendo.

Si usted ya sabe algo sobre criptografía, a continuación le mostraremos una explicación en un solo párrafo acerca de la subcapa de seguridad. Si no sabe nada, es probable que el siguiente párrafo no sea muy ilustrativo para usted (pero tal vez sea conveniente que vuelva a leerlo después de terminar el capítulo 8).

Cuando un suscriptor se conecta a una estación base, se lleva a cabo una autentificación mutua con criptografía de clave pública RSA mediante certificados X.509. Las cargas útiles mismas se cifran mediante un sistema de clave simétrica, ya sea AES (Rijndael) o DES con encadenamiento de bloques de cifrado. La verificación de integridad utiliza SHA-1. Este último párrafo no estuvo tan complicado, ¿o sí?

Ahora veamos la parte común de la subcapa MAC. Esta subcapa está orientada a conexión y de punto a multipunto, lo cual significa que una estación base se comunica con múltiples estaciones suscriptoras. Gran parte de este diseño proviene de los módems de cable, en donde un amplificador de cabecera del cable controla las transmisiones de múltiples módems de cable en las premisas de los clientes.

La dirección descendente es muy directa. La estación base controla las ráfagas de la capa física que se utilizan para enviar información a las distintas estaciones suscriptoras. La subcapa MAC simplemente empaqueta sus tramas en esta estructura. Para reducir la sobrecarga, existen varias opciones distintas. Por ejemplo, las tramas MAC se pueden enviar en forma individual, o se pueden empacar una tras otra en un grupo.

El canal ascendente es más complicado debido a que hay suscriptores que compiten por acceder a él. Su asignación está estrechamente relacionada con el aspecto de calidad del servicio. Se definen cuatro clases de servicio:

1. Servicio de tasa de bits constante.
2. Servicio de tasa de bits variable en tiempo real.
3. Servicio de tasa de bits variable en tiempo no real.
4. Servicio de mejor esfuerzo.

Todos los servicios del estándar 802.16 son orientados a conexión. Cada conexión obtiene una de las clases de servicio antes mostradas, que se determina al momento de configurar la conexión. Este diseño es muy diferente al de 802.11 o al de Ethernet, los cuales son sin conexión en la subcapa MAC.

El servicio de tasa de bits constante está diseñado para transmitir voz descomprimida. Este servicio necesita enviar una cantidad predeterminada de datos en intervalos predeterminados. Para adaptarlo se dedican ciertas ráfagas a cada conexión de este tipo. Una vez que se ha asignado el ancho de banda, las ráfagas están disponibles de manera automática, sin necesidad de solicitar cada una de ellas.

El servicio de tasa de bits variable en tiempo real está destinado para la multimedia comprimida y otras aplicaciones ligeras en tiempo real, en las que el ancho de banda necesario en cada instante puede

variar. Para adaptarlo, la estación base sondea al suscriptor con base en un intervalo fijo para saber cuánto ancho de banda se necesita esta vez.

El servicio de tasa de bits variable en tiempo no real es para las transmisiones pesadas que no son en tiempo real, como las transferencias de archivos extensos. Para este servicio, la estación base sondea al suscriptor con frecuencia, pero no a intervalos fijos prescritos. Las conexiones con este servicio pueden usar también el servicio de mejor esfuerzo (que describiremos a continuación) para solicitar ancho de banda.

El servicio de mejor esfuerzo es para todo lo demás. No se realiza sondeo y el suscriptor debe competir por el ancho de banda con otros suscriptores de mejor esfuerzo. Las solicitudes para pedir el ancho de banda se realizan en ráfagas que están marcadas en el mapa ascendente como disponibles para competir por ellas. Si una solicitud tiene éxito, su éxito se notará en el siguiente mapa descendente. Si no tiene éxito, el suscriptor desafortunado tendrá que intentar nuevamente más tarde. Para minimizar las colisiones se utiliza el algoritmo de retroceso exponencial binario de Ethernet.

4.5.5 La estructura de trama del estándar 802.16

Todas las tramas MAC comienzan con un encabezado genérico. A éste le sigue una carga útil y una suma de verificación (CRC) opcionales, como se ilustra en la figura 4-33. La carga útil no se necesita en las tramas de control; por ejemplo, en las que solicitan ranuras de canal. La suma de verificación también es (sorprendentemente) opcional, debido a la corrección de errores en la capa física y al hecho de que nunca se realiza un intento por retransmitir tramas en tiempo real. Si no se van a intentar retransmisiones, ¿para qué molestarse con una suma de verificación? Pero si hay una suma de verificación, es la CRC del estándar IEEE 802, en donde se utilizan confirmaciones de recepción y retransmisiones por cuestión de confiabilidad.

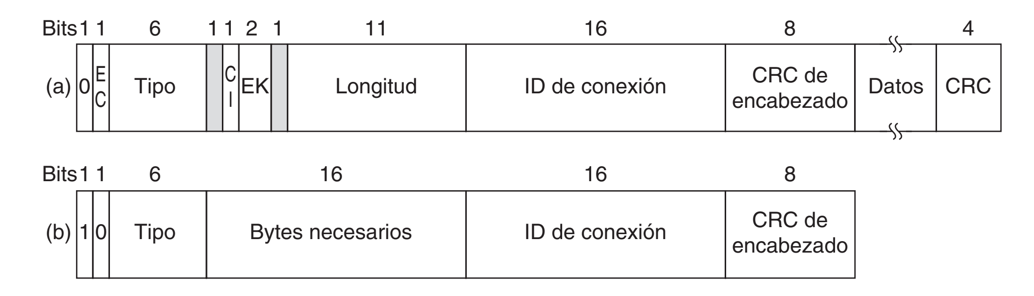


Figura 4-33. (a) Una trama genérica. (b) Una trama de solicitud de ancho de banda.

Ahora daremos un rápido vistazo a los campos de encabezado de la figura 4-33(a). El bit *EC* indica si la carga útil está cifrada. El campo *Tipo* identifica el tipo de la trama, e indica principalmente si hay empaquetamiento y fragmentación. El campo *CI* señala la presencia o ausencia de la suma de verificación final. El campo *EK* indica cuál de las claves de cifrado se está utilizando (si es que se usa alguna). El campo *Longitud* muestra la longitud exacta de la trama completa, incluyendo el encabezado. El *Identificador de conexión* señala a qué conexión pertenece esta trama. Por último, el campo *CRC de encabezado* es una suma de verificación sólo del encabezado, que utiliza el polinomio $x^8 + x^2 + x + 1$.

El protocolo 802.16 tiene muchos tipos de tramas. En la figura 4-33(b) se muestra un ejemplo de un tipo distinto de trama, que se utiliza para solicitar ancho de banda. Empieza con un bit 1 en lugar de un bit 0 y es similar al encabezado genérico, excepto que los bytes segundo y tercero forman un número de

16 bits, lo que indica cuánto ancho de banda se necesita para transmitir el número especificado de bytes. Las tramas de solicitud de ancho de banda no transportan una carga útil o una CRC de trama completa.

Podríamos decir mucho más sobre el estándar 802.16, pero no es el lugar adecuado. Para más información, consulte el estándar IEEE 802.16-2009 mismo.

4.6 BLUETOOTH

En 1994, la empresa L. M. Ericsson se interesó en conectar sus teléfonos móviles con otros dispositivos (por ejemplo, computadoras portátiles) sin necesidad de cables. En conjunto con otras cuatro empresas (IBM, Intel, Nokia y Toshiba), formó un SIG (Grupo de Interés Especial; es decir, un consorcio) en 1998 con el propósito de desarrollar un estándar inalámbrico para interconectar computadoras, dispositivos de comunicaciones y accesorios a través de radios inalámbricos de bajo consumo de energía, corto alcance y económicos. Al proyecto se le asignó el nombre **Bluetooth**, en honor de Harald Blaatand (Bluetooth) II (940-981), un rey vikingo que unificó (es decir, conquistó) Dinamarca y Noruega, también sin necesidad de cables.

El estándar Bluetooth 1.0 se liberó en julio de 1999, y desde entonces el SIG no ha vuelto su vista hacia atrás. Ahora todas las formas de dispositivos electrónicos para consumidores utilizan Bluetooth, desde los teléfonos móviles y las computadoras portátiles hasta los audífonos, impresoras, teclados, ratones, consolas de videojuegos, relojes, reproductores de música, unidades de navegación, etc. Los protocolos de Bluetooth permiten a estos dispositivos encontrarse y conectarse entre sí, a lo cual se le conoce como **emparejamiento** (*pairing*), además de que pueden transferir datos en forma segura.

Los protocolos también evolucionaron durante la última década. Después de que se estabilizaron los protocolos iniciales, se agregaron tasas de datos más altas a Bluetooth 2.0 en 2004. Con la liberación de la versión 3.0 en 2009, Bluetooth se puede usar para emparejar dispositivos junto con 802.11 para transferencia de datos a velocidades altas. La liberación de la versión 4.0 en diciembre de 2009 especificaba una operación de bajo consumo de energía. Esto será útil para las personas que no quieren cambiar las baterías con frecuencia en todos los dispositivos dispuestos alrededor del hogar. A continuación veremos los aspectos principales de Bluetooth.

4.6.1 Arquitectura de Bluetooth

Empecemos nuestro análisis del sistema Bluetooth con un rápido vistazo de lo que contiene y cuál es su propósito. La unidad básica de un sistema Bluetooth es una **piconet**, la cual consta de un nodo maestro y hasta siete nodos esclavos activos a una distancia máxima de 10 metros. Puede haber varias piconets en el mismo cuarto (grande), e incluso se pueden conectar mediante un nodo puente que participa en varias piconets, como se muestra en la figura 4-34. A una colección interconectada de piconets se le conoce como **scatternet**.

Además de los siete nodos esclavos activos en una piconet, puede haber hasta 255 nodos estacionados en la red. Éstos son dispositivos que el nodo maestro ha cambiado a un estado de bajo consumo de energía para reducir el desgaste innecesario de sus pilas. En el estado estacionado, un dispositivo no puede hacer nada excepto responder a una señal de activación o una señal baliza por parte del dispositivo maestro. También existen dos estados intermedios, *hold* y *sniff*, pero en este caso no son de nuestra incumbencia.

La razón del diseño maestro/esclavo es que los diseñadores pretendían facilitar la implementación de chips Bluetooth completos por menos de 5 dólares. La consecuencia de esta decisión es que los esclavos

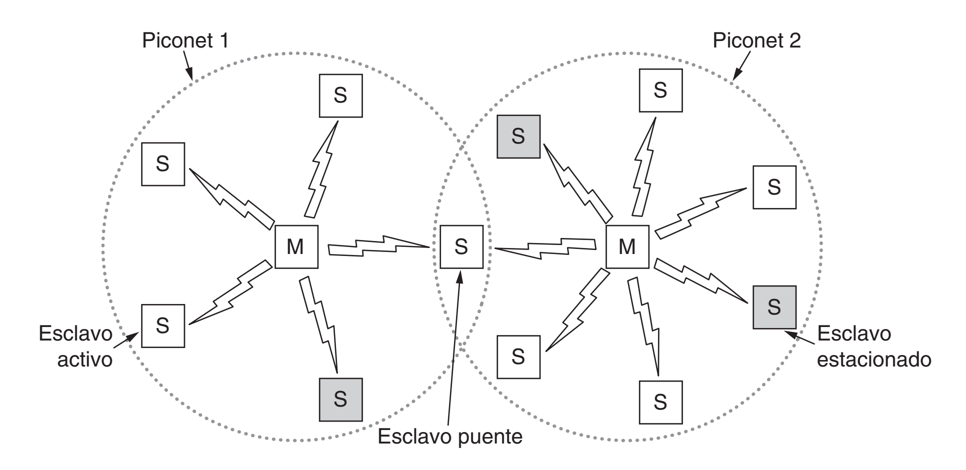


Figura 4-34. Dos piconets se pueden conectar para formar una scatternet.

son sumamente pasivos y básicamente realizan todo lo que los maestros les indican. En esencia, una piconet es un sistema TDM centralizado, en el cual el maestro controla el reloj y determina qué dispositivo se comunica en una ranura de tiempo específica. Toda la comunicación es entre el maestro y el esclavo; no es posible una comunicación directa de esclavo a esclavo.

4.6.2 Aplicaciones de Bluetooth

La mayoría de los protocolos de red sólo proporcionan canales entre las entidades que se comunican y permiten a los diseñadores de aplicaciones averiguar para qué desean utilizarlos. Por ejemplo, el estándar 802.11 no especifica si los usuarios deben utilizar sus computadoras portátiles para leer correo electrónico, navegar por web o para cualquier otro uso. En contraste, el SIG de Bluetooth especifica el soporte de aplicaciones específicas y provee distintas pilas de protocolos para cada una de ellas. Al momento de escribir este libro hay 25 aplicaciones, las cuales se denominan **perfiles**. Por desgracia, esta metodología conduce a un alto grado de complejidad. Aquí omitiremos la complejidad, pero analizaremos brevemente los perfiles para ver con más claridad lo que el SIG de Bluetooth trata de lograr.

Seis de los perfiles son para distintos usos de audio y video. Por ejemplo, el perfil intercom permite conectar dos teléfonos como *walkie-talkies*. Los perfiles (auricular) *headset* y (manos libres) *hands-free* proveen comunicación de voz entre un auricular y su estación base, y se podrían usar para la telefonía de manos libres al conducir un automóvil. Hay otros perfiles para flujo continuo de audio y video con calidad estereofónica; por ejemplo, de un reproductor de música portátil a los auriculares o de una cámara digital a una TV.

El perfil de dispositivo de interfaz humana es para conectar teclados y ratones a las computadoras. Otros perfiles permiten a un teléfono móvil u otra computadora recibir imágenes de una cámara o enviar imágenes a una impresora. Tal vez sea más interesante un perfil para usar un teléfono móvil como control remoto para una TV (habilitada para Bluetooth).

Existen otros perfiles que permiten la conexión en red. El perfil de red de área personal permite a dispositivos Bluetooth formar una red *ad hoc* o acceder en forma remota a otra red, como una LAN 802.11, por medio de un punto de acceso. El perfil de red de marcación telefónica fue de hecho la motivación original de todo el proyecto. Este perfil permite que una computadora portátil se conecte a un teléfono móvil que contenga un módem integrado sin necesidad de usar cables.

También se han definido perfiles para el intercambio de información de capas superiores. El perfil de sincronización está diseñado para cargar datos en un teléfono móvil al salir del hogar y recolectar datos de éste al momento de regresar.

Omitiremos el resto de los perfiles y sólo mencionaremos que algunos de ellos sirven como bloques básicos sobre los cuales se basan los perfiles anteriores. El perfil de acceso genérico, en el que se basan todos los demás perfiles, provee una forma de establecer y mantener enlaces (canales) seguros entre el maestro y los esclavos. Los otros perfiles genéricos definen los fundamentos del intercambio de objetos, así como el transporte de audio y video. Los perfiles utilitarios se usan mucho para funciones como emular una línea serial, que es muy útil para muchas aplicaciones heredadas.

¿Era realmente necesario explicar en detalle todas estas aplicaciones y proporcionar diferentes pilas de protocolos para cada una? Tal vez no, pero había varios grupos de trabajo distintos que diseñaron las diferentes partes del estándar, cada uno de los cuales se enfocó en su problema específico y generó su propio perfil. Piense en esto como la ley de Conway en acción (en el número de abril de 1968 de la revista *Datamation*, Melvin Conway observó que si se asignan *n* personas para escribir un compilador, se obtendrá un compilador de *n* pasos o, en forma más general, la estructura del software reflejará la estructura del grupo que la produjo). Quizás hubieran sido suficientes dos pilas de protocolos en vez de 25, una para la transferencia de archivos y otra para transmitir flujos continuos de comunicación en tiempo real.

4.6.3 La pila de protocolos de Bluetooth

El estándar Bluetooth cuenta con muchos protocolos agrupados libremente en las capas que se muestran en la figura 4-35. La primera observación que haremos es que la estructura de capas no sigue el modelo OSI, el modelo TCP/IP, el modelo 802 o algún otro modelo.

La capa inferior es la capa de radio física, la cual es bastante similar a la capa física de los modelos OSI y 802. Se encarga de la transmisión y la modulación de radio. Aquí, muchas de las cuestiones se relacionan con el objetivo de lograr que el sistema sea económico, de modo que se pueda convertir en un artículo para el mercado masivo.

La capa de control de enlace (o banda base) tiene algunos puntos en común con la subcapa MAC, pero también incluye elementos de la capa física. Se encarga de la forma en que el maestro controla las ranuras de tiempo y cómo se agrupan éstas en tramas.

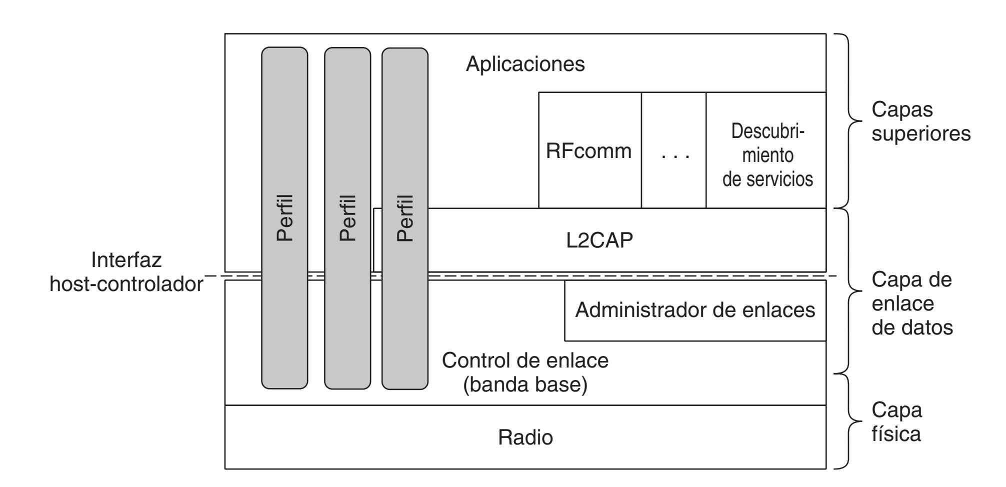


Figura 4-35. La arquitectura del protocolo Bluetooth.

A continuación se encuentran dos protocolos que usan el protocolo de control de enlace. El administrador de enlaces se encarga de establecer canales lógicos entre dispositivos, incluyendo administración de energía, emparejamiento y cifrado, así como calidad de servicio. Yace debajo de la línea de la interfaz entre host y controlador. Esta interfaz es una conveniencia para la implementación: por lo general, los protocolos debajo de la línea se implementarán en un chip Bluetooth y los protocolos arriba de la línea se implementarán en el dispositivo Bluetooth en el que se aloje el chip.

El protocolo de enlace por encima de la línea es **L2CAP** (**Protocolo de Adaptación y Control de Enlaces Lógicos**, del inglés *Logical Link Control Adaptation Protocol*). Este protocolo entrama los mensajes de longitud variable y provee confiabilidad en caso de ser necesario. Muchos protocolos utilizan L2CAP, como los dos protocolos utilitarios que se muestran. El protocolo de descubrimiento de servicios se utiliza para localizar servicios dentro de la red. El protocolo **RFcomm** (**Comunicación de Radiofrecuencia**, del inglés *Radio Frequency communication*) emula el puerto serial estándar que se encuentra en las PCs para conectar el teclado, ratón y módem, entre otros dispositivos.

En la capa superior es donde se ubican las aplicaciones. Los perfiles se representan mediante cuadros verticales debido a que cada uno define una porción de la pila de protocolos para un propósito específico. Los perfiles específicos, como el de auricular, por lo general contienen sólo los protocolos que esa aplicación necesita y ningún otro. Por ejemplo, los perfiles pueden incluir L2CAP si tienen qué enviar paquetes, pero pueden omitirlo si sólo tienen un flujo estable de muestras de audio.

En las siguientes secciones examinaremos la capa de radio de Bluetooth y varios protocolos de enlace, ya que estos corresponden más o menos con las subcapas física y MAC en las otras pilas de protocolos que hemos estudiado.

4.6.4 La capa de radio de Bluetooth

La capa de radio traslada los bits del maestro al esclavo, o viceversa. Es un sistema de baja potencia con un alcance de 10 metros que opera en la misma banda ISM de 2.4 GHz que el estándar 802.11. La banda se divide en 79 canales de 1 MHz cada uno. Para coexistir con otras redes que usan la banda ISM, se utiliza el espectro disperso de salto de frecuencia. Puede haber hasta 1600 saltos/seg sobre las ranuras con un tiempo de permanencia de 625 μseg. Todos los nodos en una piconet saltan frecuencias al mismo tiempo, siguiendo la sincronización de ranuras y la secuencia de salto pseudoaleatoria que dicta el maestro.

Por desgracia, resultó que las primeras versiones de Bluetooth y el estándar 802.11 interferían entre sí lo suficiente como para que uno arruinara las transmisiones del otro. Algunas empresas respondieron con la prohibición total de Bluetooth, pero con el tiempo se ideó una solución técnica. En esta solución, Bluetooth debe adaptar su secuencia de saltos para excluir los canales en los que haya otras señales de RF. Este proceso reduce la interferencia dañina y se le conoce como **salto de frecuencia adaptativo**.

Se utilizan tres formas de modulación para enviar bits en un canal. El esquema básico es usar la modulación por desplazamiento de frecuencia para enviar un símbolo de 1 bit cada microsegundo, con lo cual se obtienen tasas de datos brutas de 2 o 3 Mbps. Las tasas mejoradas se introdujeron con la versión 2 de Bluetooth. Estas tasas utilizan modulación por desplazamiento de fase para enviar dos o tres bits por símbolo, para tasas de datos brutas de 2 o 3 Mbps. Las tasas mejoradas son usadas sólo en la porción de datos de las tramas.

4.6.5 Las capas de enlace de Bluetooth

La capa de control de enlace (o banda base) es lo más parecido que tiene Bluetooth a una subcapa MAC. Esta capa convierte el flujo de bits puros en tramas y define algunos formatos clave. En la forma más simple, el maestro de cada piconet define una serie de ranuras de tiempo de 625 μseg; las transmisiones del maestro empiezan en las ranuras pares y las de los esclavos, en las ranuras impares. Este esquema es

la tradicional multiplexión por división de tiempo, en donde el maestro acapara la mitad de las ranuras y los esclavos comparten la otra mitad. Las tramas pueden tener 1, 3 o 5 ranuras de longitud. Cada trama tiene una sobrecarga de 126 bits para un código de acceso y encabezado, además de un tiempo de asentamiento de 250-260 μseg por salto para que los circuitos de radio económicos se estabilicen. La carga útil de la trama se puede cifrar para fines de confidencialidad con una clave que se selecciona a la hora en que se conectan el esclavo y el maestro. Los saltos sólo ocurren entre tramas, no durante una trama. El resultado es que una trama de 5 ranuras es mucho más eficiente que una trama de 1 ranura, ya que la sobrecarga es constante pero se envían más datos.

El protocolo administrador de enlaces establece canales lógicos, llamados **enlaces**, para transportar tramas entre el maestro y un dispositivo esclavo que se descubren uno al otro. Se lleva a cabo un procedimiento de emparejamiento para asegurarse que los dos dispositivos puedan comunicarse antes de usar el enlace. El viejo método de emparejamiento es que ambos dispositivos se deben configurar con el mismo NIP (Número de Identificación Personal) de cuatro dígitos. La coincidencia del NIP es la forma en que cada dispositivo sabe que se conectará al dispositivo remoto correcto. No obstante, los usuarios sin imaginación y los dispositivos recurren a valores predeterminados como “0000” y “1234” para los NIP, lo cual significa que este método provee muy poca seguridad en la práctica.

El nuevo método de **emparejamiento simple seguro** permite a los usuarios confirmar que ambos dispositivos despliegan la misma clave de contraseña, o permite observar la clave de contraseña en un dispositivo e introducirla en el segundo dispositivo. Este método es más seguro, ya que los usuarios no tienen que elegir ni establecer un NIP. Sólo confirman una clave de contraseña más extensa, generada por el dispositivo. Desde luego que este método no se puede usar en algunos dispositivos con entrada/salida limitada, como unos audífonos de manos libres.

Una vez que termina el emparejamiento, el protocolo administrador establece los enlaces. Existen dos tipos principales de enlaces para transmitir datos de usuario. El primero es el enlace **SCO** (**Síncrono Orientado a Conexión**, del inglés *Synchronous Connection Oriented*) y se utiliza para datos en tiempo real, como ocurre en las conexiones telefónicas. A este tipo de enlace se le asigna una ranura fija en cada dirección. Un esclavo puede tener hasta tres enlaces SCO con su maestro. Cada enlace SCO puede transmitir un canal de audio PCM de 64,000 bps. Debido a la naturaleza de alta prioridad de los enlaces SCO, las tramas que se envían a través de ellos nunca se retransmiten. En vez de ello se puede usar la corrección de errores hacia delante para incrementar la confiabilidad.

El otro tipo de enlace es **ACL** (**Asíncrono Sin Conexión**, del inglés *Asynchronous ConnectionLess*). Este tipo de enlace se utiliza para los datos de conmutación de paquetes que están disponibles en intervalos irregulares. El tráfico ACL se distribuye con base en el mejor esfuerzo. No se dan garantías. Se pueden perder tramas y tal vez haya que retransmitirlas. Un esclavo sólo puede tener un enlace ACL con su maestro.

Los datos que se envían sobre enlaces ACL provienen de la capa L2CAP; la cual tiene cuatro funciones principales. Primero, acepta paquetes de hasta 64 KB de las capas superiores y los descompone en tramas para su transmisión. En el otro extremo las tramas se vuelven a ensamblar en paquetes. Segundo, maneja la multiplexión y demultiplexión de paquetes de varias fuentes. Cuando se vuelve a ensamblar un paquete, determina a cuál protocolo de capa superior debe entregarlo; por ejemplo, RFcomm o descubrimiento de servicios. Tercero, maneja el control de errores y la retransmisión. Detecta los errores y reenvía los paquetes cuya recepción no se confirmó. Por último, L2CAP hace valer los requerimientos de calidad del servicio entre múltiples enlaces.

4.6.6 Estructura de la trama de Bluetooth

Bluetooth define varios formatos de trama, el más importante se muestra en dos formas en la figura 4-36. Empieza con un código de acceso que por lo general identifica al maestro, de modo que los esclavos que

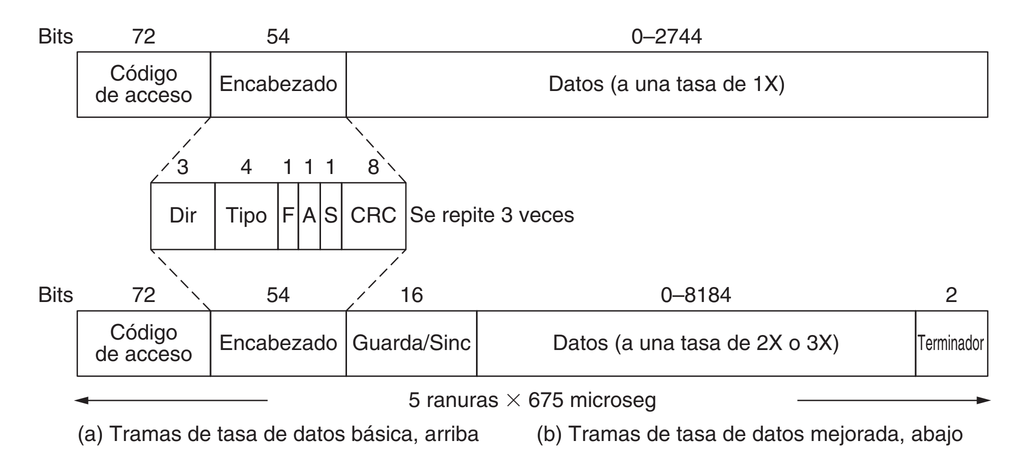


Figura 4-36. Trama de datos típica de Bluetooth con tasas de datos (a) básica y (b) mejorada.

se encuentren en el rango de alcance de dos maestros sepan cuál tráfico es para ellos. A continuación se encuentra un encabezado de 54 bits que contiene campos comunes de la subcapa MAC. Si la trama se envía a la tasa de transmisión básica, sigue el campo de datos. Éste tiene hasta 2744 bits para una transmisión de cinco ranuras. Para una sola ranura de tiempo, el formato es el mismo excepto que el campo de datos es de 240 bits.

Si la trama se envía con la tasa mejorada, la porción de datos puede tener hasta dos o tres veces la cantidad de bits, ya que cada símbolo transporta 2 o 3 bits en vez de 1 bit. A estos datos les sigue un campo de guarda y un patrón de sincronización que se utiliza para cambiar a la tasa de datos más alta. Es decir, el código de acceso y el encabezado se transportan con la tasa básica y sólo la porción de los datos se transporta con la tasa más rápida. Las tramas de tasa mejorada terminan con un terminador corto.

Demos un rápido vistazo al encabezado común. El campo *Dirección* identifica a cuál de los ocho dispositivos activos está destinada la trama. El campo *Tipo* indica el tipo de trama (ACL, SCO, de sondeo o nula), el tipo de corrección de errores que se utiliza en el campo de datos y cuántas ranuras de longitud tiene la trama. Un esclavo establece el bit *Flujo* cuando su búfer está lleno y no puede recibir más datos. Este bit permite una forma primitiva de control de flujo. El bit *A* (de confirmación de recepción) se utiliza para superponer un ACK en una trama. El bit *S* (de secuencia) se utiliza para numerar las tramas con el propósito de detectar retransmisiones. El protocolo es de parada y espera, por lo que 1 bit es suficiente. A continuación viene el encabezado *Suma de verificación* de 8 bits. Todo el encabezado de 18 bits se repite tres veces para formar el encabezado de 54 bits que se aprecia en la figura 4-36. En el lado receptor, un circuito sencillo examina las tres copias de cada bit. Si son las mismas, el bit es aceptado. De lo contrario, se impone la opinión de la mayoría. De esta forma, 54 bits de capacidad de transmisión se utilizan para enviar 10 bits de encabezado. Esto se debe a que es necesaria una gran cantidad de redundancia para enviar datos de manera confiable, en un entorno ruidoso, mediante dispositivos de bajo costo y baja potencia (2.5 mW) con poca capacidad de cómputo.

En el campo de datos de las tramas ACL y SCO se utilizan varios formatos. Las tramas SCO de tasa básica son un ejemplo simple para estudiar: el campo de datos siempre es de 240 bits. Se definen tres variantes, que permiten 80, 160 o 240 bits de carga útil real; el resto se utiliza para corrección de errores. En la versión más confiable (carga útil de 80 bits), el contenido se repite tres veces, al igual que el encabezado.

Podemos resolver la capacidad con esta trama de la siguiente forma. Como el esclavo sólo puede usar las ranuras impares, recibe 800 ranuras/seg, al igual que el maestro. Con una carga útil de 80 bits,

la capacidad de canal del esclavo es de 64 000 bps, al igual que la capacidad de canal del maestro. Esta capacidad es justo la necesaria para un solo canal de voz PCM full-dúplex (razón por la cual se eligió una tasa de saltos de 1600 saltos/seg). Esto es, a pesar de un ancho de banda puro de 1 Mbps, un solo canal de voz full-dúplex sin comprimir puede saturar la piconet por completo. La eficiencia de 13% es el resultado de invertir un 41% de la capacidad en el tiempo de establecimiento, un 20% en los encabezados y un 26% en la codificación de repetición. Esta deficiencia resalta el valor de las tasas mejoradas y las tramas de más de una ranura.

Hay mucho más que decir sobre Bluetooth, pero ya no tenemos espacio en este libro. Para los curiosos, la especificación 4.0 de Bluetooth contiene todos los detalles.

4.7 RFID

Ya vimos los diseños de MAC de las redes LAN hasta MAN, y pasamos también por las redes PAN. Como un último ejemplo, estudiaremos una categoría de dispositivos inalámbricos de gamma baja que las personas tal vez no reconozcan que también forman una red de computadoras: las etiquetas y los lectores **RFID** (**Identificación Por Radio Frecuencia**, del inglés *Radio Frequency IDentification*) que describimos en la sección 1.5.4.

La tecnología RFID toma muchas formas, puesto que se utiliza en tarjetas inteligentes, implantes para mascotas, pasaportes, libros de bibliotecas y demás. La forma que analizaremos se desarrolló durante la búsqueda de un **EPC** (**Código Electrónico de Producto**, del inglés *Electronic Product Code*) que empezó con el Auto-ID Center en el Instituto Tecnológico de Massachusetts, en 1999. Un EPC es un reemplazo para el código de barras que puede transportar una mayor cantidad de información y se puede leer vía electrónica a través de distancias de hasta 10 m, incluso cuando no es visible. Es una tecnología distinta a la de, por ejemplo, la RFID que se utiliza en los pasaportes, pues éstos se deben colocar muy cerca de un lector para realizar una transacción. Gracias a su habilidad de comunicarse a través de cierta distancia, los EPCs son más relevantes para nuestros estudios.

EPCglobal se formó en 2003 para comercializar la tecnología RFID desarrollada por el Auto-ID Center. Este esfuerzo recibió un impulso en 2005 cuando Walmart exigió a sus 100 proveedores principales etiquetar todos los envíos con etiquetas RFID. Su implementación generalizada se ha visto obstaculizada por la dificultad de competir con los códigos de barra impresos que son más económicos, pero los nuevos usos (como en las licencias de conducir) son cada vez más populares. A continuación describiremos la segunda generación de esta tecnología, conocida de manera informal como **EPC Gen 2** (EPCglobal, 2008).

4.7.1 Arquitectura EPC Gen 2

En la figura 4-37 se muestra la arquitectura de una red RFID EPC Gen 2, la cual tiene dos componentes clave: etiquetas y lectores. Las etiquetas RFID son pequeños dispositivos de bajo costo que tienen un identificador ECP único de 96 bits y una pequeña cantidad de memoria que el lector RFID puede leer y escribir. Por ejemplo, la memoria se podría usar para registrar el historial sobre la ubicación de un artículo a medida que avanza por la cadena de suministro.

Con frecuencia, las etiquetas parecen calcomanías que se pueden colocar, por ejemplo, en pares de jeans en las repisas de una tienda. Casi todo el espacio de la calcomanía lo ocupa una antena impresa sobre ella. Un pequeño punto en el centro es el circuito integrado RFID. Como alternativa, se pueden integrar las etiquetas RFID en un objeto, como una licencia de conductor. En ambos casos, las etiquetas no tienen

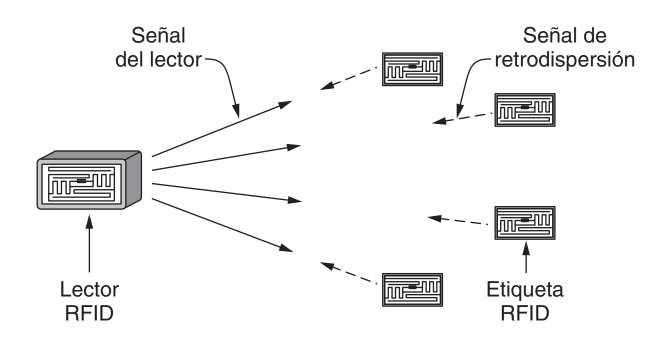


Figura 4-37. Arquitectura de RFID.

batería, por lo que deben recopilar energía de las transmisiones de radio de un lector RFID cercano para poder operar. A este tipo de etiquetas se le denomina etiqueta "clase 1", para diferenciarlas de las etiquetas con más capacidad que cuentan con baterías.

Los lectores son la inteligencia en el sistema, lo cual es análogo a las estaciones base y los puntos de acceso en las redes celulares y WiFi. Los lectores son mucho más poderosos que las etiquetas. Tienen sus propias fuentes de energía, a menudo tienen varias antenas y están a cargo de definir el momento en que las etiquetas envían y reciben mensajes. Como es común que haya varias etiquetas dentro del alcance de lectura, los lectores deben resolver el problema del múltiple acceso. También puede haber varios lectores que compitan entre sí en la misma área.

La principal tarea del lector es generar un inventario de las etiquetas en el vecindario; es decir, debe descubrir los identificadores de las etiquetas circundantes. El inventario se lleva a cabo mediante el protocolo de la capa física y el protocolo de identificación de etiquetas, los cuales describiremos en las siguientes secciones.

4.7.2 Capa física de EPC Gen 2

La capa física define cómo se envían los bits entre el lector RFID y las etiquetas. En su mayor parte se utilizan métodos para enviar señales inalámbricas que hemos visto antes. En Estados Unidos, las transmisiones se envían en la banda ISM de 902-928 MHz sin licencia. Esta banda se encuentra dentro del alcance de UHF (Frecuencia Ultra Alta), por lo que las etiquetas se denominan etiquetas RFID de UHF. El lector realiza el salto de frecuencias por lo menos cada 400 mseg para dispersar su señal a través del canal, para limitar la interferencia y cumplir con los requerimientos regulatorios. El lector y las etiquetas usan formas de modulación ASK (Modulación por Desplazamiento de Amplitud) que describimos en la sección 2.5.2 para codificar bits. Puesto que toman turnos para enviar bits, el enlace es half-dúplex.

Existen dos diferencias principales en comparación con otras capas físicas que hemos estudiado. La primera es que el lector siempre transmite una señal, sin importar que sea el lector o la etiqueta quien se está comunicando. Naturalmente, el lector transmite una señal para enviar bits a las etiquetas. Para que las etiquetas envíen bits al lector, éste transmite una señal portadora fija que no transmite bits. Las etiquetas recolectan esta señal para obtener la potencia que necesitan para operar; de no ser así, una etiqueta no podría transmitir en primer lugar. Para enviar datos, una etiqueta cambia entre la acción de reflejar la señal del lector, como una señal de radar que rebota de un objetivo, y la acción de absorberla.

A este método se le conoce como **retrodispersión**. Es distinto a todos los demás métodos inalámbricos que hemos visto hasta ahora, en donde el emisor y receptor nunca transmiten al mismo tiempo. La retrodispersión es una forma de bajo consumo de energía para que la etiqueta cree su propia señal débil y aparezca en el lector. Para que el lector decodifique la señal entrante, debe filtrar la señal de salida que

está transmitiendo. Como la señal de la etiqueta es débil, las etiquetas sólo pueden enviar bits al lector a una tasa baja, y no pueden recibir ni detectar transmisiones de otras etiquetas.

La segunda diferencia es que se utilizan formas muy simples de modulación, de modo que se puedan implementar en una etiqueta que opere con muy poca potencia y su costo de fabricación sea de unos cuantos centavos. Para enviar datos a las etiquetas, el lector usa dos niveles de amplitud. Los bits se determinan para que sean un 0 o un 1, dependiendo de cuánto tiempo espere el lector antes de un periodo de baja potencia. La etiqueta mide el tiempo entre los periodos de baja potencia y compara este tiempo con una referencia que se mide durante un preámbulo. Como se muestra en la figura 4-38, los 1s son más largos que los 0s.

Las respuestas de las etiquetas consisten en que la etiqueta alterne su estado de retrodispersión a intervalos fijos, para crear una serie de pulsos en la señal. Se pueden usar desde uno hasta ocho periodos de pulsos para codificar cada 0 o 1, dependiendo del grado de confiabilidad que se requiera. Los 1s tienen menos transiciones que los 0s, como se muestra mediante un ejemplo de codificación de un periodo de dos pulsos en la figura 4-38.

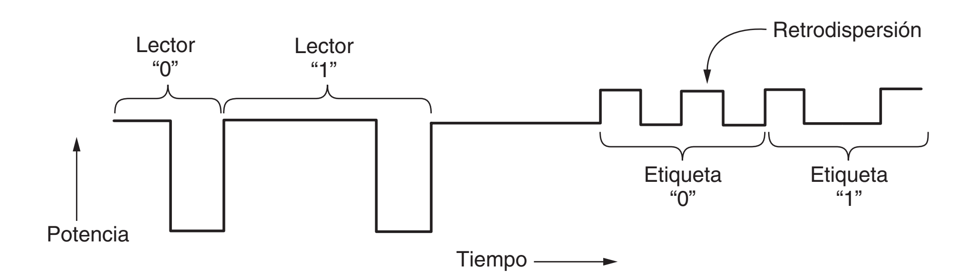


Figura 4-38. Señales de retrodispersión de etiquetas y del lector.

4.7.3 Capa de identificación de etiquetas de EPC Gen 2

Para generar un inventario de las etiquetas circundantes, el lector necesita recibir un mensaje de cada etiqueta, en el cual se proporcione la identificación de la misma. Esta situación es un problema de acceso múltiple, en el cual se desconoce el número de etiquetas en el caso general. El lector podría difundir una consulta para pedir a todas las etiquetas que envíen sus identificadores. Sin embargo, las etiquetas que respondieran de inmediato tendrían una colisión en forma muy parecida a las colisiones de las estaciones en una red Ethernet clásica.

En este capítulo ya hemos visto muchas formas de lidiar con el problema de acceso múltiple. El protocolo más apropiado para la situación actual, en la que las etiquetas no pueden escuchar las transmisiones de las demás, es el ALOHA ranurado, uno de los primeros protocolos que estudiamos. Este protocolo se adaptó para usarlo en el RFID Gen 2.

En la figura 4-39 se muestra la secuencia de mensajes usados para identificar una etiqueta. En la primera ranura (ranura 0), el lector envía un mensaje *Query* (consulta) para iniciar el proceso. Cada mensaje *QRepeat* avanza a la siguiente ranura. El lector también indica a las etiquetas el rango de ranuras a través del cual deben aleatorizar las transmisiones. Es necesario usar un rango, ya que el lector sincroniza las etiquetas al iniciar el proceso; a diferencia de las estaciones en una red Ethernet, las etiquetas no despiertan con un mensaje en el momento que deseen.

Las etiquetas seleccionan una ranura al azar para responder. En la figura 4-39, la etiqueta responde en la ranura 2. Sin embargo, las etiquetas no envían sus identificadores cuando responden por primera

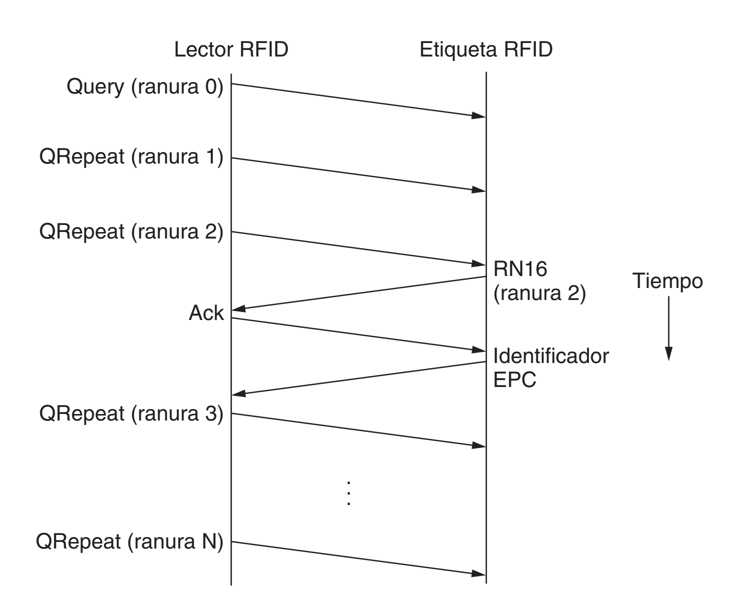


Figura 4-39. Ejemplo de intercambio de mensajes para identificar una etiqueta.

vez. En cambio, una etiqueta envía un número aleatorio corto de 16 bits en un mensaje *RN16*. Si no hay colisión, el lector recibe este mensaje y envía un mensaje *ACK* por su cuenta. En esta etapa, la etiqueta ha adquirido la ranura y envía su identificador EPC.

La razón de este intercambio es que los identificadores EPC son largos, por lo que las colisiones en estos mensajes serían muy caras. En cambio, se usa un intercambio corto para probar si la etiqueta puede usar en forma segura la ranura para enviar su identificador. Una vez que se transmite con éxito su identificador, la etiqueta deja temporalmente de responder a los nuevos mensajes *Query* de modo que se puedan identificar las etiquetas restantes.

Un problema clave es que el lector ajuste el número de ranuras para evitar colisiones, pero sin usar tantas ranuras como para que se vea afectado el desempeño. Este ajuste es análogo al retroceso exponencial binario en Ethernet. Si el lector ve demasiadas ranuras sin respuestas, o demasiadas ranuras con colisiones, puede enviar un mensaje *QAdjust* para reducir o aumentar el rango de ranuras a través de las cuales responden las etiquetas.

El lector RFID puede realizar otras operaciones en las etiquetas. Por ejemplo, puede seleccionar un subconjunto de etiquetas antes de llevar a cabo un inventario, lo cual le permite recolectar respuestas, por decir, de jeans etiquetados pero no de camisas etiquetadas. El lector también puede escribir datos en las etiquetas al momento en que se identifican. Esta característica se podría usar para registrar el punto de venta o cualquier otra información relevante.

4.7.4 Formatos de los mensajes de identificación de etiquetas

En la figura 4-40 se muestra el formato del mensaje *Query* como un ejemplo de un mensaje de lector a etiqueta. El mensaje es compacto puesto que las tasas descendentes son limitadas, de 27 kbps a 128 kbps. El campo *Comando* transmite el código 1000 para identificar el mensaje como un *Query*.

Las siguientes banderas, *DR*, *M* y *TR*, determinan los parámetros de la capa física para las transmisiones del lector y las respuestas de las etiquetas. Por ejemplo, la tasa de respuestas se puede establecer en un valor entre 5 kbps y 640 kbps. Omitiremos los detalles de estas banderas.

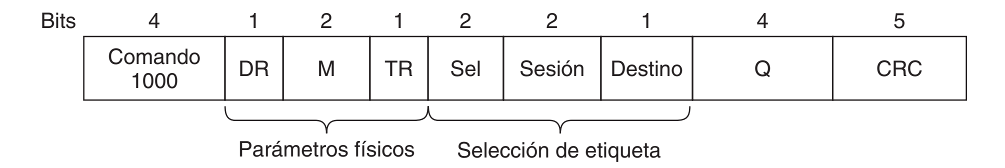


Figura 4-40. Formato del mensaje Query.

Después siguen tres campos, *Sel*, *Sesión* y *Destino*, los cuales seleccionan las etiquetas para responder. Así como los lectores pueden seleccionar un subconjunto de identificadores, las etiquetas mantienen el registro de hasta cuatro sesiones concurrentes y si se identificaron o no en esas sesiones. De esta forma, varios lectores pueden operar en áreas de cobertura que se traslapen, mediante el uso de distintas sesiones.

A continuación viene el parámetro más importante para este comando, *Q*. Este campo define el rango de ranuras a través de las cuales responderán las etiquetas, de 0 a $2^Q - 1$. Por último, hay una CRC para proteger los campos del mensaje. Con 5 bits, es más corta que la mayoría de las CRC que hemos visto, pero también el mensaje *Query* es mucho más corto que la mayoría de los paquetes.

Los mensajes de etiqueta a lector son más simples. Como el lector tiene el control, sabe qué mensaje esperar en respuesta a cada una de sus transmisiones. Las respuestas de las etiquetas sólo transmiten datos, como el identificador EPC.

En un principio las etiquetas eran sólo para fines de identificación. No obstante, han crecido con el tiempo y se asemejan a computadoras muy pequeñas. Algunas etiquetas de investigación tienen sensores y son capaces de ejecutar pequeños programas para recopilar y procesar datos (Sample y colaboradores, 2008). Una visión de esta tecnología es la “Internet de cosas”, que conecta a Internet objetos en el mundo físico (Welbourne y colaboradores, 2009; y Gershenfield y colaboradores, 2004).

4.8 CONMUTACIÓN DE LA CAPA DE ENLACE DE DATOS

Muchas organizaciones tienen varias redes LAN y desean interconectarlas. ¿No sería conveniente si tan sólo pudiéramos unir las redes LAN para formar una LAN más grande? De hecho, este tipo de redes se puede conectar mediante dispositivos llamados **puentes**. Los switches de Ethernet que describimos en la sección 4.3.4 son un nombre moderno para los puentes; proveen una funcionalidad que va más allá de los hubs de Ethernet clásica y Ethernet para facilitar la unión de varias redes LAN en una red más grande y veloz. Utilizaremos los términos “puente” y “switch” para indicar lo mismo.

Los puentes operan en la capa de enlace de datos, por lo que examinan las direcciones de la capa de enlace de datos para reenviar tramas. Como no tienen que examinar el campo de carga útil de las tramas que reenvían, pueden manejar paquetes IP al igual que otros tipos de paquetes, como AppleTalk. En contraste, los *enrutadores* examinan las direcciones de los paquetes y realizan su trabajo de enrutamiento con base en ellas, por lo que sólo funcionan con los protocolos para los cuales se diseñaron.

En esta sección analizaremos la forma en que funcionan los puentes y cómo se utilizan para unir varias redes LAN físicas en una sola LAN lógica. También veremos cómo hacer lo inverso y tratar una LAN física como varias redes LAN lógicas, llamadas redes **VLAN** (**LAN virtuales**, del inglés *Virtual LANs*). Ambas tecnologías proveen una flexibilidad conveniente para administrar redes. Para un análisis más completo sobre puentes, switches y temas relacionados, consulte a Seifert y Edwards (2008), y a Perlman (2000).

4.8.1 Usos de los puentes

Antes de entrar de lleno a la tecnología de los puentes, veamos algunas situaciones comunes en las cuales se utilizan los puentes. Mencionaremos tres razones por las cuales una sola organización podría terminar trabajando con varias LAN.

En primer lugar, muchas universidades y departamentos corporativos tienen sus propias redes LAN para conectar sus propias computadoras personales, servidores y dispositivos como impresoras. Dado que los objetivos de los distintos departamentos difieren, los distintos departamentos pueden establecer diferentes redes LAN, sin importarles lo que hagan los demás departamentos. Pero tarde o temprano surge la necesidad de interacción, y aquí es donde entran los puentes. En este ejemplo surgieron múltiples redes LAN debido a la autonomía de sus propietarios.

En segundo lugar, la organización puede estar distribuida geográficamente en varios edificios, separados por distancias considerables. Puede ser más económico tener redes LAN independientes en cada edificio y conectarlas mediante puentes y unos cuantos enlaces de fibra óptica de larga distancia que tender todos los cables hacia un solo switch central. Incluso si es fácil tender los cables, existen límites en cuanto a sus longitudes (por ejemplo, 200 m para Gigabit Ethernet de par trenzado). La red no funcionaría con cables más largos debido a la excesiva atenuación de la señal, o al retardo de viaje redondo. La única solución es dividir la LAN e instalar puentes para unir las piezas y poder incrementar la distancia física total que se puede cubrir.

En tercer lugar, tal vez sea necesario dividir lo que por lógica es una sola LAN en varias redes LAN individuales (conectadas mediante puentes) para manejar la carga. Por ejemplo, en muchas universidades grandes, hay miles de estaciones de trabajo disponibles para los estudiantes y el cuerpo docente. Las empresas también pueden tener miles de empleados. La escala de este sistema hace imposible poner todas las estaciones de trabajo en una sola LAN; hay muchas más computadoras que puertos en cualquier hub Ethernet y más estaciones de lo que se permite en una sola Ethernet clásica.

Incluso si fuera posible cablear todas las estaciones de trabajo juntas, al colocar más estaciones en un hub Ethernet o en una red Ethernet clásica no se agrega capacidad. Todas las estaciones comparten la misma cantidad fija de ancho de banda. Entre más estaciones haya, menor será el ancho de banda promedio por estación.

Sin embargo, dos redes LAN separadas tienen el doble de la capacidad de una sola LAN. Los puentes permiten unir redes LAN y mantener al mismo tiempo esta capacidad. La clave es no enviar tráfico a los puertos en los que no se necesita, de modo que cada LAN pueda operar a toda velocidad. Este comportamiento también aumenta la confiabilidad, ya que en una sola LAN, un nodo defectuoso que siga transmitiendo un flujo continuo de basura puede llegar a obstruir toda la LAN completa. Al decidir qué reenviar o no, los puentes actúan como puertas contra incendios en un edificio, pues evitan que un solo nodo errático haga fallar todo el sistema.

Para que estos beneficios pudieran estar fácilmente disponibles, los puentes ideales tendrían que ser totalmente transparentes. Debería ser posible comprar los puentes, conectar los cables de LAN en los puentes y que todo funcionara a la perfección en un instante. No debería existir la necesidad de cambios de hardware o de software, ni de configurar switches de direcciones o descargar tablas de enrutamiento o parámetros, nada de eso. Simplemente conectar los cables y seguir con nuestras actividades cotidianas. Lo que es más, la operación de las redes LAN existentes no se debería ver afectada por los puentes para nada. En cuanto a las estaciones, no debería haber ninguna diferencia observable en cuanto a si son parte o no de una LAN con puente. Debería ser igual de fácil mover estaciones alrededor de una LAN con puente que moverlas en una sola LAN.

Aunque resulta sorprendente, en realidad es posible crear puentes que sean transparentes. Se utilizan dos algoritmos: un algoritmo de aprendizaje hacia atrás para detener el tráfico que se envía a donde no

es necesario, y un algoritmo de árbol de expansión para romper los ciclos que se pueden formar cuando los switches se conectan entre sí de manera no intencional. Ahora analizaremos cada uno de estos algoritmos para ver cómo se logra esta magia.

4.8.2 Puentes de aprendizaje

La topología de dos redes LAN conectadas por un puente se muestra en la figura 4-41 para dos casos. En el lado izquierdo, dos redes LAN multiderivación (por ejemplo, redes Ethernet clásicas) se unen mediante una estación especial (el puente) que se sitúa entre ambas redes LAN. Del lado derecho, se unen redes LAN con cables punto a punto, incluyendo un hub. Los puentes son los dispositivos a los que se conectan las estaciones y el hub. Si la tecnología de LAN es Ethernet, los puentes son mejor conocidos como switches Ethernet.

Los puentes se desarrollaron cuando se usaban redes Ethernet clásicas, por lo que a menudo se muestran en topologías con cables multiderivación, como en la figura 4-41(a). Sin embargo, todas las topologías en la actualidad están compuestas de cables punto a punto y switches. Los puentes funcionan de la misma forma en ambas configuraciones. Todas las estaciones conectadas al mismo puerto en un puente pertenecen al mismo dominio de colisión, y éste es distinto al dominio de colisión para otros puertos. Si hay más de una estación, como en una red Ethernet clásica, un hub o un enlace half-dúplex, se utiliza el protocolo CSMA/CD para enviar tramas.

Sin embargo, hay una diferencia en cuanto a la forma en que se construyen las redes LAN con puentes. Para conectar redes LAN multiderivación con puentes, se agrega un puente como una nueva estación en cada LAN multiderivación, como en la figura 4-41(a). Para conectar redes LAN punto a punto mediante puentes, los hubs se conectan a un puente o, lo que es preferible, se reemplazan con un puente para incrementar el desempeño. En la figura 4-41(b) los puentes reemplazaron a todos los hubs excepto uno.

También se pueden conectar distintos tipos de cables a un puente. Por ejemplo, el cable que conecta el puente *B1* con el puente *B2* en la figura 4-41(b) podría ser un enlace de fibra óptica de larga distancia, mientras que el cable que conecta los puentes con las estaciones podría ser una línea de par trenzado de corta distancia. Esta disposición es útil para conectar redes LAN mediante puentes en distintos edificios.

Ahora consideremos lo que ocurre dentro de los puentes. Cada puente opera en modo promiscuo; es decir, acepta cada una de las tramas que transmiten las estaciones conectadas a cada uno de sus puertos. El

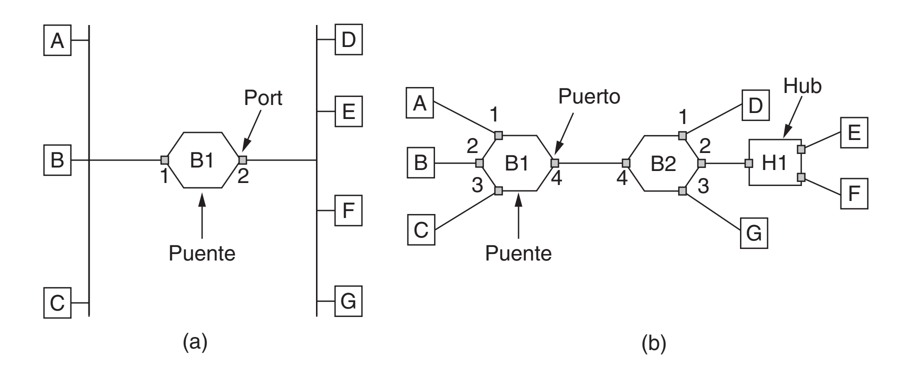


Figura 4-41. (a) Puente que conecta dos redes LAN multiderivación. (b) Puentes (y un hub) que conectan siete estaciones punto a punto.

puente debe decidir si va a reenviar o desechar cada trama y, en caso de que sea la primera opción, también debe decidir por qué puerto enviar la trama. Esta decisión se basa en la dirección de destino. Como ejemplo, considere la topología de la figura 4-41(a). Si la estación *A* envía una trama a la estación *B*, el puente *B1* recibirá la trama en el puerto 1. Esta trama se puede desechar de inmediato sin más preámbulos, debido a que ya se encuentra en el puerto correcto. Sin embargo, suponga que en la topología de la figura 4-41(b) la estación *A* envía una trama a *D*. El puente *B1* recibirá la trama en el puerto 1 y la enviará por el puerto 4. Después el puente *B2* recibirá la trama en su puerto 4 y la enviará por el puerto 1.

Una forma simple de implementar este esquema es mediante una gran tabla (hash) dentro del puente. La tabla puede listar cada posible destino y a qué puerto de salida pertenece. Por ejemplo, en la figura 4-41(b), la tabla en *B1* listaría a *D* como perteneciente al puerto 4, ya que todo lo que *B1* tiene que saber es por qué puerto enviar las tramas para llegar a *D*. El que, de hecho, se lleven a cabo más reenvíos posteriormente cuando la trama llegue a *B2* no es de interés para *B1*.

Cuando se conectan por primera vez los puentes, todas las tablas de hash están vacías. Ninguno de los puentes sabe dónde se encuentran los destinos, por lo que utilizan un algoritmo de inundación: todas las tramas que llegan con un destino desconocido se envían por todos los puertos a los que está conectado el puente, excepto por el que llegaron. Con el paso del tiempo, los puentes aprenden dónde están los destinos. Una vez conocido un destino, las tramas destinadas para él se colocan sólo en el puerto apropiado; no se inundan.

El algoritmo que usan los puentes es el de **aprendizaje hacia atrás**. Como ya mencionamos, los puentes funcionan en modo promiscuo y de esta manera pueden ver todas las tramas que se envían por cualquiera de sus puertos. Al analizar las direcciones de origen, pueden saber cuáles máquinas están disponibles en cuáles puertos. Por ejemplo, si el puente *B1* de la figura 4-41(b) ve una trama en el puerto 3 que proviene de *C*, sabe que es posible acceder a *C* por medio del puerto 3, así que registra una entrada en su tabla de hash. Cualquier trama subsecuente dirigida a *C* que llegue desde el puente *B1* por cualquier puerto se reenviará al puerto 3.

La topología puede cambiar conforme las máquinas y los puentes se enciendan y apaguen, o cuando se trasladan de un sitio a otro. Para manejar topologías dinámicas, siempre que se realiza una entrada en una tabla de hash se registra en la entrada la hora de llegada de una trama. Cada vez que llega una trama cuyo origen ya está en la tabla, su entrada se actualiza con la hora actual. Así, la hora asociada a cada entrada indica la última vez que se registró una trama proveniente de esa máquina.

Un proceso en el puente analiza de manera periódica la tabla de hash y purga todas las entradas que tengan más de algunos minutos de antigüedad. De esta manera, si una computadora se desconecta de su LAN, se traslada a otro lugar del edificio y se vuelve a conectar en algún otro lugar, en pocos minutos volverá a funcionar con normalidad, sin necesidad de intervención manual. Este algoritmo también significa que si una máquina está inactiva durante algunos minutos, el tráfico destinado a ella se inundará hasta que la máquina misma envíe una trama.

El procedimiento de enrutamiento para una trama entrante depende del puerto por el que llegue (el puerto de origen) y de la dirección a la cual está destinada (la dirección de destino). El procedimiento se muestra a continuación.

1. Si el puerto para la dirección de destino es el mismo que el puerto de origen, se desecha la trama.
2. Si el puerto para la dirección y el puerto de origen son diferentes, se reenvía la trama por el puerto de destino.
3. Si se desconoce el puerto de destino, se recurre a la inundación y envía la trama por todos los puertos excepto el de origen.

Tal vez se pregunte si el primer caso puede ocurrir con enlaces punto a punto. La respuesta es que puede ocurrir si se usan hubs para conectar un grupo de computadoras a un puente. En la figura 4-41(b) se

muestra un ejemplo, en donde las estaciones *E* y *F* se conectan al hub *H* 1, el cual a su vez está conectado al puente *B2*. Si *E* envía una trama a *F*, el hub la retransmitirá a *B* y también a *F*. Eso es lo que hacen los hubs; conectan todos los puertos entre sí, de modo que la trama que entre por un puerto simplemente se envíe por todos los demás puertos. La trama llegará a *B2* en el puerto 4, que ya es el puerto de salida correcto para llegar al destino. El puente *B2* sólo tiene que desechar la trama.

A medida que llega cada trama es necesario aplicar este algoritmo, por lo que con frecuencia se implementa mediante chips VLSI de propósito especial. Los chips realizan la búsqueda y actualizan la entrada en la tabla, todo en unos cuantos microsegundos. Como los puentes sólo analizan las direcciones MAC para decidir cómo reenviar las tramas, es posible empezar a reenviar tan pronto como llega el campo del encabezado de destino, antes de que haya llegado el resto de la trama (siempre y cuando la línea de salida esté disponible, claro está). Este diseño reduce la latencia de pasar a través del puente, así como el número de tramas que el puente debe ser capaz de colocar en el búfer. Se denomina **conmutación al vuelo** (*cut-through*) o **enrutamiento de agujero de gusano** (*wormhole*) y por lo general se maneja en el hardware.

Podemos ver la operación de un puente en términos de las pilas de protocolos, para comprender lo que significa ser un dispositivo de capa de enlace. Considere una trama que se envió de la estación *A* a la estación *D* en la configuración de la figura 4-41(a), en donde las redes LAN son Ethernet. La trama pasará a través de un puente. La vista de procesamiento de la pila de protocolos se muestra en la figura 4-42.

El paquete llega de una capa superior y desciende a la capa MAC Ethernet. Adquiere un encabezado de Ethernet (y también un terminador, que no se muestra en la figura). Esta unidad se pasa a la capa física, sale por el cable y el puente la recoge.

En el puente, la trama se pasa de la capa física a la capa MAC Ethernet. Esta capa tiene un procesamiento extendido, en comparación con la capa MAC en una estación. Pasa la trama a un retransmisor, todavía dentro de la capa MAC. La función de retransmisión del puente sólo usa el encabezado MAC Ethernet para determinar cómo manejar la trama. En este caso, pasa la trama a la capa MAC Ethernet del puerto utilizado para llegar a la estación *D*, y la trama continúa su camino.

En el caso general, la retransmisión en una capa dada puede rescribir los encabezados para esa capa. En breve veremos un ejemplo con redes VLAN. En ningún caso el puente deberá ver dentro de la trama y descubrir que transmite un paquete IP; esto es irrelevante para el procesamiento del puente y violaría el uso de capas de protocolos. Cabe mencionar además que un puente con *k* puertos tendrá *k* instancias de capas MAC y físicas. El valor de *k* es 2 para nuestro simple ejemplo.

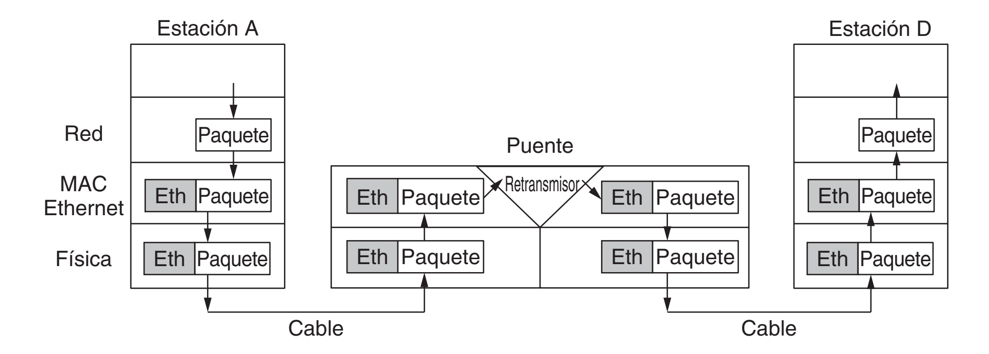


Figura 4-42. Procesamiento de protocolos en un puente.

4.8.3 Puentes con árbol de expansión

Para incrementar la confiabilidad, se pueden usar enlaces redundantes entre puentes. En el ejemplo de la figura 4-43, hay dos enlaces en paralelo entre un par de puentes. Este diseño asegura que si se corta un enlace, la red no se dividirá en dos conjuntos de computadoras que no se pueden comunicar entre sí.

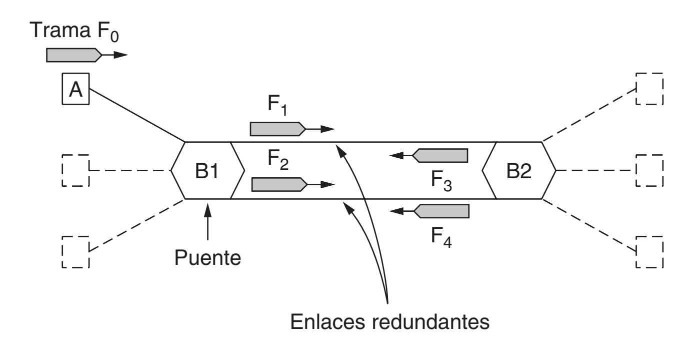


Figura 4-43. Puentes con dos enlaces paralelos.

Sin embargo, esta redundancia introduce algunos problemas adicionales, porque crea ciclos en la topología. En la figura 4-43 podemos ver un ejemplo simple de estos problemas, al observar cómo se maneja la forma en que la estación *A* envía una trama a un destino que no se había observado antes. Cada puente sigue la regla normal para el manejo de destinos desconocidos, que es inundar la trama. Sea F_0 la trama de *A* que llega al puente *B1*. El puente envía copias de esta trama a todos sus otros puertos. Sólo consideraremos los puertos del puente que conectan *B1* con *B2* (aunque la trama se enviará también a los demás puertos). Como hay dos enlaces de *B1* a *B2*, dos copias de la trama llegarán a *B2*. En la figura 4-43 se muestran como F_1 y F_2.

Poco después, el puente *B2* recibe estas tramas. Sin embargo, no sabe (ni puede saber) que son copias de la misma trama, en vez de ser dos tramas distintas que se envían una después de la otra. Por lo tanto, el puente *B2* toma la trama F_1 y envía copias de ella a todos los demás puertos; además toma a F_2 y envía copias de ella a todos los otros puertos. Esto produce las tramas F_3 y F_4 que se envían a través de los dos enlaces, de vuelta a *B1*. A continuación, el puente *B1* ve dos nuevas tramas con destinos desconocidos y las copia de nuevo. Este ciclo continúa en forma indefinida.

La solución a este problema es que los puentes se comuniquen entre sí y cubran la topología existente con un árbol de expansión que llegue a todos los puentes. En efecto, algunas conexiones potenciales entre los puentes se ignoran en el afán de construir una topología ficticia libre de ciclos, que sea un subconjunto de la topología actual.

Por ejemplo, en la figura 4-44 vemos cinco puentes interconectados y que también tienen estaciones conectadas. Cada estación se conecta sólo a un puente. Hay algunas conexiones redundantes entre los puentes, de modo que las tramas se reenviarán en ciclos si se utilizan todos los enlaces. Podemos considerar esta topología como un grafo en el que los puentes son los nodos y los enlaces punto a punto son los bordes. El grafo se puede reducir a un árbol de expansión, el cual no tiene ciclos por definición, si eliminamos los enlaces que se muestran como líneas punteadas en la figura 4-44. Si usamos este árbol de expansión, hay exactamente una ruta de cada estación a cada una de las demás estaciones. Una vez que los puentes se hayan puesto de acuerdo en cuanto al árbol de expansión, todos los reenvíos entre las estaciones se hacen a través del árbol de expansión. Puesto que existe una única ruta de cada origen a cada destino, es imposible que se produzcan ciclos.

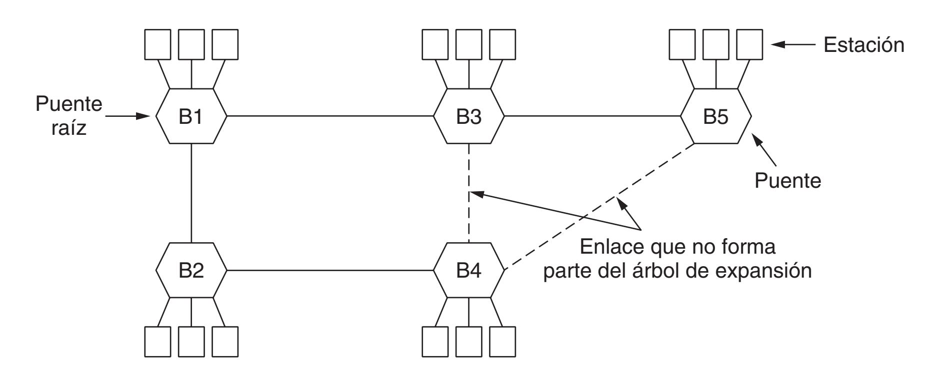


Figura 4-44. Un árbol de expansión que conecta cinco puentes. Las líneas punteadas son enlaces que no forman parte del árbol de expansión.

Para construir el árbol de expansión, los puentes ejecutan un algoritmo distribuido. Cada puente difunde en forma periódica un mensaje de configuración a través de todos sus puertos hacia sus vecinos y procesa los mensajes que recibe de otros puentes, como veremos a continuación. Estos mensajes no se reenvían, ya que su propósito es construir el árbol y usarlo para los reenvíos.

Los puentes primero tienen que escoger una conexión que sea la raíz del árbol de expansión. Para hacer esta elección, cada uno incluye un identificador basado en su dirección MAC en el mensaje de configuración, junto con el identificador del puente que creen que es la raíz. El fabricante instala las direcciones MAC, que tienen la garantía de ser únicas en todo el mundo, por lo cual son identificadores convenien-tes y únicos. Los puentes seleccionan la conexión con el menor identificador para que sea la raíz. Después de haber intercambiado suficientes mensajes para esparcir las noticias, todos los puentes se pondrán de acuerdo en cuál será la raíz. En la figura 4-44, el puente *B1* tiene el menor identificador y se convierte en la raíz.

A continuación se construye un árbol con las rutas más cortas de la raíz a cada uno de los puentes. En la figura 4-44 se puede llegar a los puentes *B2* y *B3* directamente desde el puente *B1*, en un salto que sea una de las rutas más cortas. Se puede llegar al puente *B4* en dos saltos, a través de *B2* o de *B3*. Para romper este empate se selecciona la ruta por el puente con el menor identificador, así que se llega a *B4* a través de *B2*. Se puede llegar al puente *B5* en dos saltos a través de *B3*.

Para encontrar estas rutas más cortas, los puentes incluyen la distancia desde la raíz en sus mensajes de configuración. Cada puente recuerda la ruta más corta que encuentra hacia la raíz. Después, los puentes desactivan los puertos que no formen parte de la ruta más corta.

Aunque el árbol abarca todos los puentes, no todos los enlaces (e incluso los puentes) están necesariamente presentes en el árbol. Esto ocurre debido a que al desactivar los puertos se cortan algunos enlaces de la red para evitar los ciclos. Incluso después de que se ha establecido el árbol de expansión, el algoritmo continúa en ejecución durante la operación normal con el fin de detectar de manera automática los cambios en la topología y actualizar el árbol.

El algoritmo para construir el árbol de expansión fue inventado por Radia Perlman. Su trabajo era resolver el problema de unir redes LAN sin ciclos. Se le dio una semana para hacerlo, pero se le ocurrió la idea del algoritmo del árbol de expansión en un día. Por fortuna le quedó el tiempo suficiente para escribirlo como poema (Perlman, 1985):

Creo que nunca veré
Un grafo más encantador que un árbol.
Un árbol cuya propiedad crucial

Es la conectividad sin ciclos.
Un árbol que debe ser capaz de extenderse.
Para que los paquetes puedan llegar a cada LAN.
Primero hay que seleccionar la raíz
Se elije mediante ID.
Se trazan las rutas de menor costo desde la raíz
En el árbol se establecen estas rutas.
La gente como yo hace una red
Y después los puentes encuentran un árbol de expansión.

Después, el algoritmo del árbol de expansión se estandarizó como el IEEE 802.1D y se usó durante muchos años. En 2001 se revisó para encontrar con más rapidez un nuevo árbol de expansión después de un cambio de topología. Para un análisis detallado de los puentes, consulte a Perlman (2000).

4.8.4 Repetidores, hubs, puentes, switches, enrutadores y puertas de enlace (gateways)

Hasta ahora hemos visto una variedad de formas para desplazar tramas y paquetes de una computadora a otra. Hemos mencionado repetidores, hubs, puentes, switches, enrutadores y puertas de enlace. Todos estos dispositivos son de uso común, aunque difieren en formas sutiles y no tan sutiles. Puesto que son tantos, tal vez valga la pena analizarlos en conjunto para conocer sus similitudes y diferencias.

La clave para entender estos dispositivos es tener en cuenta que operan en distintas capas, como se ilustra en la figura 4-45(a). La capa es importante porque los distintos dispositivos utilizan diferentes piezas de información para decidir cómo van a conmutar. En un escenario común, el usuario genera algunos datos para enviarlos a una máquina remota. Estos datos se pasan a la capa de transporte, que le agrega un encabezado (por ejemplo, un encabezado TCP) y pasa la unidad que resulta a la capa de red. Ésta le agrega su propio encabezado para formar un paquete de capa de red (por ejemplo, un paquete IP). En la figura 4-45(b) podemos ver el paquete IP sombreado en color gris. Después, el paquete pasa a la capa de enlace de datos, la cual agrega su propio encabezado y una suma de verificación (CRC), y entrega la trama resultante a la capa física para su transmisión; por ejemplo, sobre una LAN.

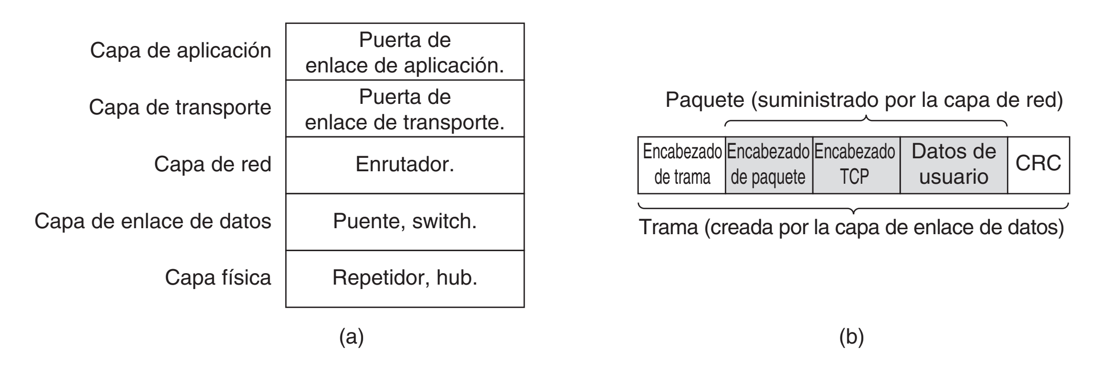


Figura 4-45. (a) Qué dispositivo está en cada capa. (b) Tramas, paquetes y encabezados.

Ahora veamos los dispositivos de conmutación y cómo se relacionan con los paquetes y las tramas. En la parte inferior (en la capa física) se encuentran los repetidores. Éstos son dispositivos analógicos que funcionan con señales de los cables a los que están conectados. Una señal que aparece en un cable

se limpia, amplifica y pone en otro cable. Los repetidores no distinguen entre tramas, paquetes o encabezados. Ellos comprenden los símbolos que codifican bits como voltios. Por ejemplo, la Ethernet clásica se diseñó para permitir cuatro repetidores que aumentaran la señal para extender la longitud máxima de cable de 500 a 2 500 metros.

Pasemos ahora a los hubs. Un hub tiene varias líneas de entrada que unen de manera eléctrica. Las tramas que llegan a cualquiera de las líneas se envían por todas las demás. Si dos tramas llegan al mismo tiempo colisionarán, al igual que en un cable coaxial. Todas las líneas que convergen en un hub deben operar a la misma velocidad. A diferencia de los repetidores, los hubs (por lo general) no amplifican las señales entrantes y están diseñados para múltiples líneas de entrada, aunque las diferencias son ligeras. Al igual que los repetidores, los hubs son dispositivos de capa física que no examinan las direcciones de la capa de enlace ni las utilizan de ninguna manera.

Veamos a continuación la capa de enlace de datos, en donde se encuentran los puentes y los switches. Ya hemos visto algo de los puentes. Un puente conecta dos o más redes LAN. Al igual que un hub, un puente moderno cuenta con múltiples puertos, por lo general suficientes para tener de 4 a 48 líneas de entrada de cierto tipo. A diferencia de un hub, cada puerto está aislado para ser su propio domino de colisión; si el puerto tiene una línea punto a punto full-dúplex, no se necesita el algoritmo CSMA/CD. Cuando llega una trama, el puente extrae la dirección de destino del encabezado y la busca en una tabla para averiguar a dónde debe enviar la trama. En Ethernet, esta dirección es la dirección de destino de 48 bits que se muestra en la figura 4-14. El puente sólo envía la trama por el puerto en el que se necesita y puede reenviar varias tramas al mismo tiempo.

Los puentes ofrecen un desempeño muy superior al de los hubs, además el aislamiento entre los puertos del puente también significa que las líneas de entrada pueden operar a distintas velocidades, e incluso tal vez con distintos tipos de redes. Un ejemplo común es un puente con puertos que se pueden conectar a redes Ethernet de 10, 100 y 1000 Mbps. Se requiere un búfer dentro del puente para aceptar una trama en un puerto y transmitirla por un puerto distinto. Si las tramas llegan con más rapidez de lo que se pueden retransmitir, el puente se puede quedar sin espacio de búfer y tal vez tenga que empezar a desechar tramas. Por ejemplo, si una red Gigabit Ethernet está transmitiendo bits a una red Ethernet de 10 Mbps a máxima velocidad, el puente tendrá que colocar las tramas en un búfer y ver si no se queda sin memoria. Este problema existe aunque todos los puertos operen a la misma velocidad, ya que tal vez varios puertos envíen tramas a un puerto de destino dado.

Los puentes se diseñaron originalmente para poder unir distintos tipos de redes LAN; por ejemplo, una LAN Ethernet y una LAN Token Ring. Sin embargo, esto nunca funcionó bien debido a las diferencias entre las redes LAN. En los distintos formatos de trama se requieren procesos de copia y reformateo, para lo cual se necesita tiempo de CPU, es necesario calcular una nueva suma de verificación además de que se introduce la posibilidad de que haya errores sin detectar debido a bits defectuosos en la memoria del puente. Las distintas longitudes máximas de trama también son un problema grave sin una buena solución. En esencia, se deben desechar las tramas que son demasiado largas como para reenviarlas. Adiós a la transparencia.

Otras dos áreas en las que las redes LAN pueden diferir son la seguridad y la calidad del servicio. Algunas redes LAN tienen cifrado en la capa de enlace (por ejemplo, 802.11) y otras no (por ejemplo, Ethernet). Algunas redes LAN tienen características de calidad de servicio tales como las prioridades (por ejemplo, 802.11) y otras no (como Ethernet). En consecuencia, cuando una trama debe viajar entre estos tipos de LAN, tal vez no se pueda proveer la seguridad o calidad del servicio que espera el emisor. Por todas estas razones, los puentes modernos funcionan por lo general con un tipo de red, y los enrutadores (que veremos en breve) son los que se usan para unir redes de distintos tipos.

Los switches son otro nombre para los puentes modernos. Las diferencias se relacionan más con la comercialización que con las cuestiones técnicas, aunque hay algunos puntos que vale la pena conocer.

Los puentes se desarrollaron cuando se usaba la Ethernet clásica, ellos tendían a unir pocas redes LAN y, por ende, tienen pocos puertos. Hoy en día es más popular el término "switch". Además, todas las instalaciones modernas usan enlaces punto a punto, como los cables de par trenzado, por lo que cada computadora se conecta directamente a un switch y es lógico que tenga muchos puertos. Por último, "switch" también se utiliza como un término general. Con un puente, la funcionalidad es clara. Por otro lado, un switch se puede referir a un switch Ethernet o a un tipo de dispositivo por completo diferente que toma decisiones de reenvío, como un conmutador telefónico (switch telefónico).

Hasta ahora hemos visto repetidores y hubs, que en realidad son bastante similares, así como puentes y switches, que son aún más similares. Ahora pasaremos a los enrutadores, que son distintos de todos los anteriores componentes. Cuando un paquete llega a un enrutador, se quita el encabezado y el terminador de la trama, y se pasa el campo de carga útil de la trama (sombreado en la figura 4-45) al software de enrutamiento. Este software usa el encabezado del paquete para elegir una línea de salida. En un paquete IP, el encabezado contiene una dirección de 32 bits (IPv4) o 128 bits (IPv6), pero no una dirección IEEE 802 de 48 bits. El software de enrutamiento no ve las direcciones de las tramas y ni siquiera sabe si el paquete llegó por una LAN o por una línea punto a punto. En el capítulo 5 estudiaremos los enrutadores y el enrutamiento.

Una capa más arriba tenemos las puertas de enlace de transporte. Estos dispositivos conectan dos computadoras que utilizan diferentes protocolos de transporte orientados a conexión. Por ejemplo, imagine que una computadora que utiliza el protocolo TCP/IP orientado a conexión necesita comunicarse con una computadora que emplea un protocolo distinto de transporte orientado a conexión, el cual se conoce como SCTP. La puerta de enlace de transporte puede copiar los paquetes de una conexión a la otra y darles el formato que necesiten.

Por último, las puertas de enlace de aplicación entienden el formato y contenido de los datos; y pueden traducir los mensajes de un formato a otro. Por ejemplo, una puerta de enlace de correo electrónico puede traducir los mensajes de Internet en mensajes SMS para teléfonos móviles. Al igual que "switch", "puerta de enlace" es algo así como un término general. Se refiere a un proceso de reenvío que opera en una capa alta.

4.8.5 Redes LAN virtuales

En los primeros días de las redes de área local, cables amarillos gruesos serpenteaban por los ductos de muchos edificios de oficinas. Conectaban a todas las computadoras por las que pasaban. No importaba cuál computadora pertenecía a cuál LAN. Todos los usuarios de oficinas cercanas se conectaban a la misma LAN aunque no estuvieran relacionados con ella. La geografía triunfaba sobre los gráficos organizacionales corporativos.

Todo cambió con el surgimiento de los cables de par trenzado y los hubs en la década de 1990. El cableado de los edificios se renovó (a un costo considerable) para desechar todas las mangueras amarillas de jardín e instalar cables de par trenzado desde cada oficina hasta gabinetes centrales al final de cada pasillo o en una sala central de máquinas, como se observa en la figura 4-46. Si el vicepresidente a cargo del cableado era un visionario, se instalaba cable de par trenzado categoría 5; si era un simple administrador, se instalaba el cable telefónico (categoría 3) existente (que tenía que reemplazarse algunos años más tarde con la aparición de Fast Ethernet).

En la actualidad, los cables han cambiado y los hubs se han convertido en switches, pero el patrón de cableado sigue siendo el mismo. Este patrón hace posible la configuración de redes LAN lógicas en vez de físicas. Por ejemplo, si una empresa desea *k* redes LAN, podría comprar *k* switches. Al elegir con cuidado qué conectores enchufar en qué switches, los ocupantes de una LAN se pueden seleccionar de tal forma que tenga sentido organizacional, sin tener mucho en cuenta la geografía.

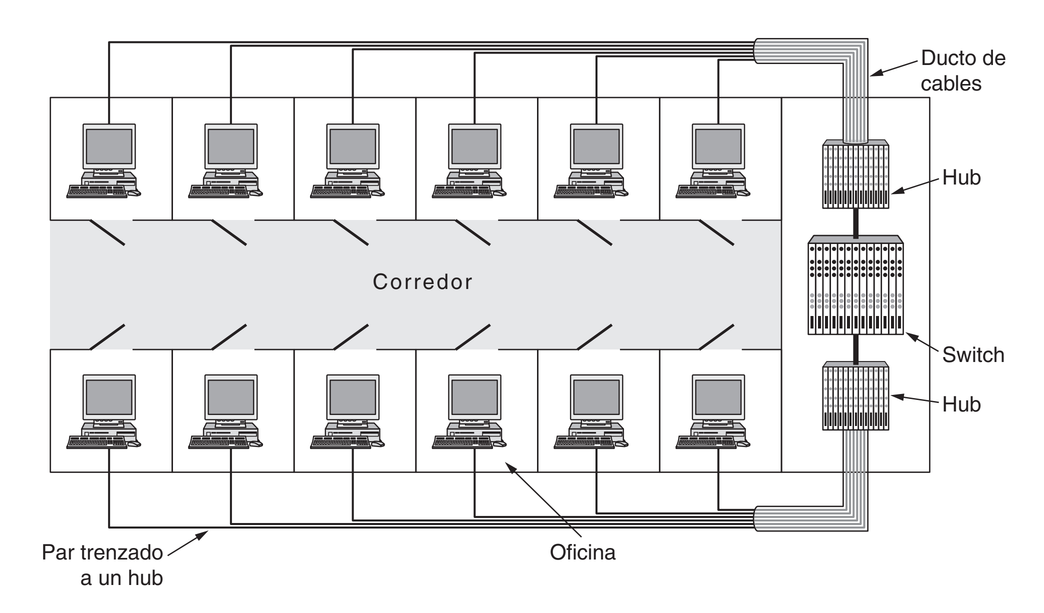


Figura 4-46. Un edificio con cableado centralizado en el que se usan hubs y un switch.

¿Es importante quién está en qué LAN? Después de todo, en casi todas las organizaciones las redes LAN están interconectadas. En resumen, por lo general sí es importante. Por diversas razones, a los administradores de red les gusta agrupar a los usuarios en redes LAN para reflejar la estructura de la orga-nización más que el diseño físico del edificio. Un aspecto es la seguridad. Una LAN podría hospedar a los servidores web y otras computadoras destinadas para uso público. Otra LAN podría hospedar a las computadoras que contengan los registros del departamento de recursos humanos que no deben salir de ese departamento. En este caso, se justifica que todas las computadoras se asignen a una sola LAN y que no se permita acceder a los servidores fuera de esa LAN. A los directivos no les agrada escuchar que un arreglo de este tipo es imposible.

Un segundo aspecto es la carga. Algunas redes LAN se utilizan mucho más que otras, y en ocasiones podría ser conveniente separarlas. Por ejemplo, si los usuarios de investigaciones realizan toda clase de experimentos que en ocasiones se les van de las manos y saturan su LAN, tal vez a los usuarios de contabilidad no les agrade tener que ceder parte de su capacidad que usaban en las videoconferencias para ayudarles. Por otra parte, esto podría infundir en la gerencia la necesidad de instalar una red más veloz.

Un tercer aspecto es el tráfico de difusión. Los puentes difunden el tráfico cuando no conocen la ubicación de destino, y los protocolos de capas superiores también usan la difusión. Por ejemplo, cuando un usuario desea enviar un paquete a una dirección IP *x*, ¿cómo sabe qué dirección MAC poner en la trama? En el capítulo 5 estudiaremos este asunto, pero en pocas palabras, la respuesta es que debe difundir una trama con la pregunta: "¿Quién posee la dirección IP *x*?" y esperar la respuesta. A medida que aumenta el número de computadoras en una LAN, también aumenta el número de difusiones. Cada difusión consume más capacidad de la LAN que una trama regular, ya que se entrega a todas las computadoras en la LAN. Al evitar que las redes LAN crezcan más de lo necesario, se reduce el impacto del tráfico de difusión.

Las difusiones tienen el problema asociado de cuando una interfaz de red se avería o desconfigura y empieza a generar flujos interminables de tramas de difusión. Si la red es realmente mala, algunas de estas tramas provocarán respuestas que a su vez generarán más tráfico. El resultado de esta **tormenta**

de difusión es que (1) las tramas de difusión ocupan toda la capacidad de la LAN, y (2) las máquinas de todas las redes LAN interconectadas se atascan con sólo procesar y desechar todas las tramas difundidas.

A primera vista parecería que podemos limitar la magnitud de las tormentas de difusión si separamos las redes LAN mediante puentes o switches, pero si el objetivo es conseguir transparencia (es decir, que una máquina se pueda cambiar a una LAN distinta al otro lado del puente sin que nadie lo note), entonces los puentes tienen que reenviar las tramas difundidas.

Ya que analizamos por qué las empresas podrían requerir varias redes LAN con alcances limitados, regresemos al problema de desacoplar la topología lógica de la física. Para construir una topología física que refleje la estructura organizacional tal vez se requiera más trabajo y aumente el costo, incluso con un cableado centralizado y switches. Por ejemplo, si dos personas en el mismo departamento trabajan en distintos edificios, puede ser más fácil conectarlos a distintos switches que pertenezcan a redes LAN diferentes. Aun si éste no es el caso, un usuario podría transferirse de un departamento a otro de la misma empresa sin cambiar de oficina, o podría cambiar de oficina pero no de departamento. Así, el usuario podría estar en la LAN incorrecta hasta que un administrador cambiara el conector del usuario de un switch a otro. Además, tal vez el número de computadoras que pertenecen a distintos departamentos no sea adecuado para el número de puertos en los switches; algunos departamentos podrían ser demasiado pequeños y otros tan grandes que requieran varios switches. Como resultado, se desperdician los puertos que no se utilizan de los switches.

En muchas empresas, los cambios organizacionales ocurren todo el tiempo, lo cual quiere decir que los administradores de sistemas desperdician mucho tiempo quitando y metiendo conectores de un lado a otro. Asimismo, en algunos casos el cambio no se puede realizar de ninguna manera porque el cable de par trenzado de la máquina del usuario está demasiado lejos del switch correcto (por ejemplo, en otro edificio), o los puertos disponibles del switch están en la LAN incorrecta.

En respuesta a la demanda de mayor flexibilidad por parte de los usuarios, los fabricantes de redes empezaron a trabajar en una forma de volver a cablear edificios completos mediante software. El concepto que surgió se denomina **VLAN** (**LAN Virtual**). El comité IEEE 802 lo estandarizó y ahora se ha implementado ampliamente en muchas organizaciones. Ahora vamos a analizarlo de forma breve. Si desea información adicional sobre redes VLAN, consulte a Seifert y Edwards (2008).

Las redes VLAN se basan en switches especialmente diseñados para este propósito. Para configurar una red VLAN, el administrador de la red decide cuántas VLAN habrá, qué computadoras habrá en cuál VLAN y cómo se llamarán las VLAN. A menudo se les asignan nombres mediante colores (de manera informal), ya que de esta manera es posible imprimir diagramas a color que muestren la disposición física de las máquinas, con los miembros de la LAN roja en rojo, los de la LAN verde en verde, etc. De esta forma, tanto el diseño físico como el lógico se pueden reflejar en un solo esquema.

Por ejemplo, considere la LAN con puente de la figura 4-47, en la cual nueve de las máquinas pertenecen a la VLAN G (gris) y cinco forman parte de la VLAN W (blanca). Las máquinas de la VLAN gris están distribuidas a través de dos switches, incluyendo dos máquinas que se conectan a un switch mediante un hub.

Para que las VLAN funcionen correctamente, es necesario establecer tablas de configuración en los puentes. Estas tablas indican cuáles VLAN se pueden acceder a través de qué puertos. Cuando una trama llega procedente de, digamos, la VLAN gris, se debe reenviar a todos los puertos identificados con una G. Esto es válido para el tráfico ordinario (es decir, de unidifusión) en el que los puentes no conocen la ubicación del destino, así como para el tráfico de multidifusión y de difusión. Cabe mencionar que podemos etiquetar un puerto con varios colores de VLAN

Como ejemplo, suponga que una de las estaciones grises conectadas al puente *B1* en la figura 4-47 envía una trama a un destino que no se conoce de antemano. El puente *B1* recibirá la trama y verá que proviene de una máquina en la VLAN gris, por lo que inundará esa trama en todos los puertos etiquetados

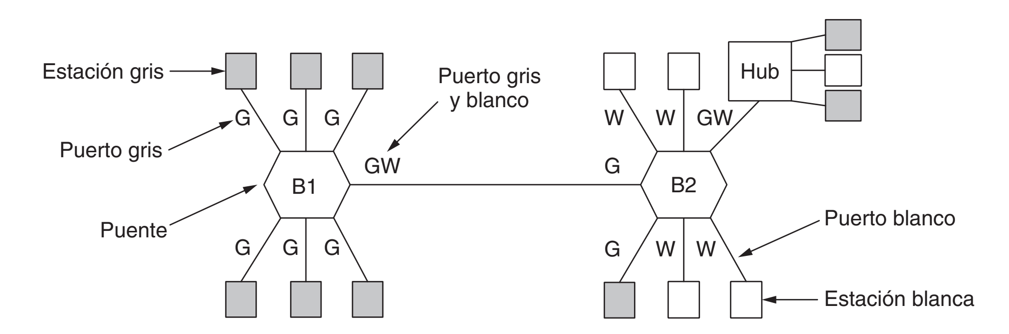


Figura 4-47. Dos redes VLAN, gris y blanca, en una LAN con puente.

como G (excepto el puerto entrante). La trama se enviará a las otras cinco estaciones grises conectadas a *B1*, así como a través del enlace de *B1* al puente *B2*. En el puente *B2*, la trama se reenvía de manera similar a todos los puertos etiquetados como G. Esto envía la trama a una estación más y al hub (que transmitirá la trama a todas sus estaciones). El hub tiene ambas etiquetas debido a que se conecta a las máquinas de ambas redes VLAN. La trama no se envía en otros puertos que no tengan G en la etiqueta, puesto que el puente sabe que no hay máquinas en la VLAN gris a las que se pueda llegar por medio de estos puertos.

En nuestro ejemplo, la trama sólo se envía del puente *B1* al puente *B2* ya que hay máquinas en la VLAN gris que están conectadas a *B2*. Si analizamos la VLAN blanca, podemos ver que el puerto del puente *B2* que se conecta al puente *B1 no* está etiquetado como W. Esto significa que una trama en la VLAN blanca no se reenviará del puente *B2* al puente *B1*. Este comportamiento es correcto, ya que no hay estaciones en la VLAN blanca que estén conectadas a *B1*.

El estándar IEEE 802.1Q

Para implementar este esquema, los puentes necesitan saber a qué VLAN pertenece una trama entrante. Sin esta información, por ejemplo, cuando el puente *B2* recibe una trama del puente *B1* en la figura 4-47, no puede saber si reenviar la trama a la VLAN gris o blanca. Si estuviéramos diseñando un nuevo tipo de LAN, sería muy fácil sólo agregar un campo VLAN en el encabezado. Pero, ¿qué podemos hacer con Ethernet, que es la LAN dominante y no tiene campos disponibles para el identificador VLAN?

El comité IEEE 802 se enfrentó a este problema en 1995. Después de muchas discusiones, hizo lo impensable y cambió el encabezado de Ethernet. El nuevo formato se publicó en el estándar IEEE **802.1Q**, emitido en 1998. El nuevo formato contiene una etiqueta VLAN, que examinaremos en breve. No es de sorprender que cambiar algo tan bien establecido como el encabezado de Ethernet no sea nada sencillo. Algunas de las preguntas que nos vienen a la mente son:

1. ¿Tenemos que tirar a la basura los cientos de millones de tarjetas Ethernet existentes?
2. Si no es así, ¿quién generará los nuevos campos?
3. ¿Qué sucederá con las tramas que ya tienen el tamaño máximo?

Por supuesto que el comité 802 estaba (aunque con mucho desagrado) consciente de estos problemas y tenía que encontrar soluciones, lo cual hizo.

La clave para la solución consiste en comprender que los campos VLAN sólo los utilizan los puentes y los conmutadores, *no* las máquinas de los usuarios. Así, en la figura 4-47 no es realmente necesario que

estén presentes en las líneas que van hacia las estaciones finales, siempre y cuando se encuentren en la línea entre los puentes. Además, para utilizar VLAN, los puentes deben tener soporte para VLAN. Este hecho ayuda a que el diseño sea viable.

Respecto a la cuestión de si es necesario desechar todas las tarjetas Ethernet existentes, la respuesta es no. Recuerde que el comité 802.3 no pudo conseguir que la gente cambiara el campo *Tipo* por un campo *Longitud*. Ya podrá imaginar la reacción ante el anuncio de que todas las tarjetas Ethernet existentes tuvieran que desecharse. Sin embargo, las nuevas tarjetas Ethernet son compatibles con el 802.1Q y pueden llenar bien los campos VLAN.

Puesto que puede haber computadoras (y switches) que no tengan soporte para VLAN, el primer puente con soporte para VLAN en tocar una trama agrega campos VLAN y el último en el camino los elimina. En la figura 4-48 se muestra un ejemplo de una topología mixta. En esta figura, las computadoras con soporte para VLAN generan tramas etiquetadas (802.1Q) directamente, y los switches posteriores utilizan estas etiquetas. Los símbolos sombreados tienen soporte para VLAN y los vacíos no.

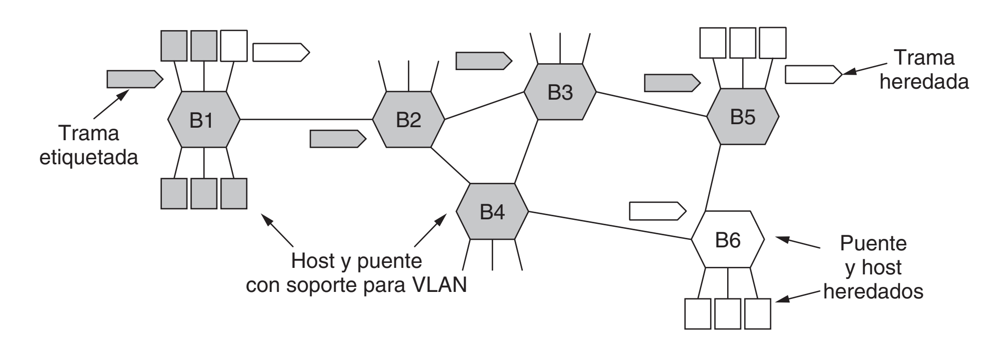


Figura 4-48. LAN con puentes que sólo cuenta con soporte parcial para VLAN. Los símbolos sombreados tienen soporte para VLAN. Los vacíos no.

En el 802.1Q, se asignan colores a las tramas dependiendo del puerto por el que se reciban. Para que este método funcione, todas las máquinas en un puerto deben pertenecer a la misma VLAN, lo cual reduce la flexibilidad. Por ejemplo, en la figura 4-48 esta propiedad es válida para todos los puertos en donde se conecte una computadora individual a un puente, pero no para el puerto en donde el hub se conecta al puente *B2*.

Además, el puente puede usar el protocolo de la capa superior para seleccionar el color. De esta forma, las tramas que llegan a un puerto se podrían colocar en distintas redes VLAN, dependiendo de si transmiten paquetes IP o tramas PPP.

Hay otros métodos posibles, pero no están soportados por el estándar 802.1Q. Como ejemplo, se puede usar la dirección MAC para seleccionar el color de VLAN. Esto podría ser útil para tramas que provienen de una LAN 802.11 cercana, en donde las computadoras portátiles envían tramas a través de distintos puertos a medida que se desplazan. En este caso, una dirección MAC se asignaría a una VLAN fija, sin importar por qué puerto entró a la LAN.

En cuanto al problema de las tramas mayores a 1518 bytes, el 802.1Q tan sólo incrementó el límite a 1522 bytes. Por suerte, sólo las computadoras y switches con soporte para VLAN deben soportar estas tramas más largas.

Ahora veamos el formato de trama del 802.1Q, que se muestra en la figura 4-49. El único cambio es la adición de un par de campos de 2 bytes. El primero es *ID del protocolo de VLAN*. Siempre tiene el valor 0x8100. Como este número es mayor de 1500, todas las tarjetas Ethernet lo interpretan como un tipo y no

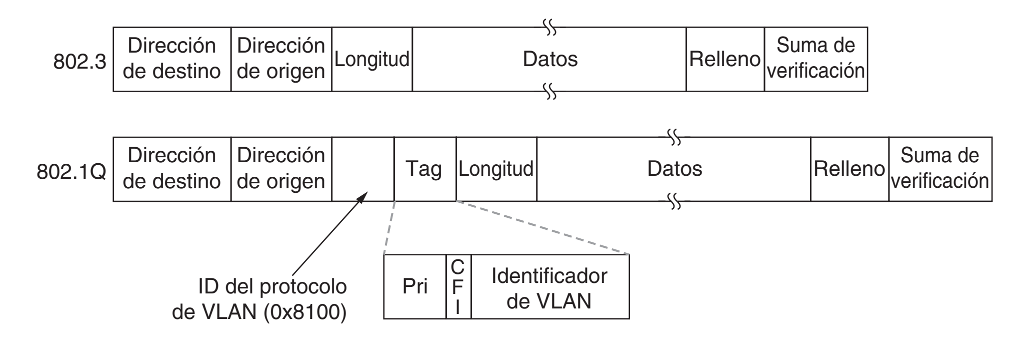


Figura 4-49. Los formatos de trama Ethernet 802.3 (heredada) y 802.1Q.

como una longitud. Lo que una tarjeta heredada hace con una trama como ésta es discutible, ya que dichas tramas no deberían enviarse a tarjetas heredadas.

El segundo campo de 2 bytes contiene tres subcampos. El principal es *Identificador de VLAN*, que ocupa los 12 bits de menor orden. Éste es el punto central de la cuestión: el color de la VLAN a la que pertenece la trama. El campo *Prioridad* de 3 bits no tiene absolutamente nada que ver con las VLAN, pero como cambiar el encabezado Ethernet es un suceso poco frecuente que tarda tres años y ocupa a un ciento de personas, ¿por qué no incorporarle algunas otras cosas buenas en el proceso? Este campo permite distinguir el tráfico en tiempo real estricto del tráfico en tiempo real flexible y del tráfico insensible al tiempo, con el propósito de ofrecer una mejor calidad de servicio sobre Ethernet. Esto es necesario para el transporte de voz sobre Ethernet (aunque, con toda franqueza, el IP tiene un campo similar desde hace un cuarto de siglo y nadie lo utiliza).

El último campo, *CFI* (*Indicador del Formato Canónico*), debió haberse llamado *CEI* (*Indicador del ego corporativo*). Su propósito original era indicar el orden de los bits en las direcciones MAC (*little endian* en comparación con *big endian*), pero su uso se perdió en otras controversias. En la actualidad, su presencia indica que la carga útil contiene una trama 802.5 congelada-seca que espera encontrar otra LAN 802.5 en el destino, cuando se transmita a través de Ethernet. Por supuesto, este arreglo no tiene absolutamente nada que ver con las VLAN. Pero la política de los comités de estándares no difiere mucho de la política común: si votas por mi bit, votaré por el tuyo.

Como ya mencionamos, cuando una trama etiquetada llega a un switch con soporte para VLAN, éste utiliza el identificador de la VLAN como índice en una tabla para averiguar a cuáles puertos enviar la trama. Pero, ¿de dónde proviene la tabla? Si se construye en forma manual, tenemos que empezar desde cero: la configuración manual de los puentes. La ventaja de los puentes transparentes es que son *plug-and-play* y no requieren configuración manual. Sería una gran pena perder esa propiedad. Por fortuna, los puentes con soporte para VLAN también se pueden autoconfigurar con sólo observar las etiquetas entrantes. Si una trama etiquetada como VLAN 4 llega por el puerto 3, entonces aparentemente una máquina en el puerto 3 se encuentra en la VLAN 4. El estándar 802.1Q explica cómo construir las tablas de manera dinámica, en su mayor parte haciendo referencia a porciones apropiadas del estándar 802.1D.

Antes de abandonar el tema del enrutamiento para VLAN, vale la pena hacer una última observación. Mucha gente en el mundo de Internet y Ethernet es fanática de las redes sin conexión y se oponen por completo a todo lo que huela a conexiones en las capas de enlace de datos y de red. No obstante, las redes VLAN incluyen un aspecto que es muy similar a una conexión. Para utilizar las VLAN de manera apropiada, cada trama lleva un identificador especial nuevo que se utiliza como índice en una tabla dentro del switch para averiguar el destino al que se debe enviar la trama. Esto es lo que se hace en las redes orientadas a conexión. En las redes sin conexión, la dirección de destino es la que se utiliza para el enrutamiento, no un tipo de identificador de conexión. En el capítulo 5 veremos más sobre este conexionismo gradual.

4.9 RESUMEN

Algunas redes tienen un solo canal que se usa para todas las comunicaciones. En estas redes, el aspecto clave del diseño es la asignación del canal entre las estaciones competidoras que desean usarlo. FDM y TDM son esquemas simples y eficientes de asignación cuando el número de estaciones es pequeño y fijo, y con el tráfico continuo. Ambos esquemas se usan ampliamente en estas circunstancias; por ejemplo, para dividir el ancho de banda de las troncales telefónicas. Sin embargo, cuando el número de estaciones es grande y variable, o la mayoría del tráfico es en ráfagas (el caso común en las redes de computadoras), FDM y la TDM no son opciones adecuadas.

Se han ideado numerosos algoritmos de asignación de canales dinámicos. El protocolo ALOHA con y sin ranuras se utiliza en muchos sistemas reales; por ejemplo, los módems de cable y RFID. Como una mejora cuando se puede detectar el estado del canal, las estaciones pueden evitar que comience una transmisión mientras otra estación está transmitiendo. Esta técnica conocida como detección de portadora ha producido varios protocolos CSMA que se pueden usar en LAN y MAN. Es la base para las redes Ethernet clásica y 802.11.

Hay una clase muy conocida de protocolos que eliminan por completo la contención, o cuando menos la reducen de manera considerable. El protocolo de mapa de bits, las topologías como los anillos y el protocolo de conteo binario descendente eliminan por completo la contención. El protocolo de recorrido de árbol la reduce al dividir en forma dinámica las estaciones en dos grupos separados de distintos tamaños y al permitir la contención sólo dentro de un grupo; lo ideal es elegir ese grupo de modo que sólo una estación esté lista para enviar cuando se le permita hacerlo.

Las LAN inalámbricas tienen dos problemas adicionales: es difícil detectar las transmisiones que colisionan y las regiones de cobertura de las estaciones pueden diferir. En la LAN inalámbrica dominante, IEEE 802.11, las estaciones usan CSMA/CA para mitigar el primer problema al dejar pequeños huecos para evitar colisiones. Las estaciones también pueden usar el protocolo RTS/CTS para combatir las terminales ocultas que surgen debido al segundo problema. Lo común es usar el IEEE 802.11 para conectar computadoras portátiles y otros dispositivos a puntos de acceso inalámbricos, aunque también se puede usar entre dispositivos. Se puede usar una de varias capas físicas, incluyendo FDM multicanal con y sin varias antenas, además de espectro disperso.

Al igual que el 802.11, los lectores y las etiquetas RFID usan un protocolo de acceso aleatorio para comunicar los identificadores. Otras redes PAN y MAN inalámbricas tienen diseños diferentes. El sistema Bluetooth conecta auriculares y muchos tipos de periféricos a las computadoras sin necesidad de cables. El IEEE 802.16 provee un servicio de datos de Internet inalámbrico de área amplia para computadoras fijas y móviles. Ambas redes usan un diseño centralizado, orientado a conexión, en donde el maestro de Bluetooth y la estación base WiMAX deciden cuándo puede cada estación enviar o recibir datos. Para el 802.16, este diseño soporta diferentes calidades de servicio para el tráfico en tiempo real, como las llamadas telefónicas, y para el tráfico interactivo, como la navegación web. Para Bluetooth, al dejar la complejidad en el maestro se pueden producir dispositivos esclavos económicos.

Ethernet es la forma dominante de LAN alámbrica. La Ethernet clásica usaba CSMA/CD para la asignación de canales en un cable amarillo del tamaño de una manguera de jardinería que serpenteaba de una máquina a otra. La arquitectura ha cambiado a medida que las velocidades han ido aumentando de 10 Mbps a 10 Gbps y continúan su ascenso. Ahora, los enlaces de punto a punto como el par trenzado se conectan a hubs y switches. Con los switches modernos y los enlaces full-dúplex, no hay contención en los enlaces y el switch puede reenviar tramas entre distintos puertos en paralelo.

Con edificios llenos de LAN, se necesita una manera de interconectarlas. Para este fin se utilizan los puentes tipo *plug-and-play*, los cuales incluyen un algoritmo de aprendizaje hacia atrás y un algoritmo

de árbol de expansión. Ya que esta funcionalidad se incluye en los switches modernos, se utilizan los términos "puente" y "switch" de manera indistinta. Par ayudar con la administración de las redes LAN con puentes, las VLAN permiten dividir la topología física en distintas topologías lógicas. El estándar de VLAN, IEEE 802.1Q, introduce un nuevo formato para las tramas Ethernet.

PROBLEMAS

1. Para este problema, utilice una fórmula de este capítulo, pero primero enúnciela. Las tramas llegan en forma aleatoria a un canal de 100 Mbps para su transmisión. Si el canal está ocupado cuando llega una trama, ésta espera su turno en una cola. La longitud de la trama se distribuye exponencialmente con un promedio de 10 000 bits/trama. Para cada una de las siguientes tasas de llegada de tramas, obtenga el retardo experimentado por la trama promedio, incluyendo tanto el tiempo de encolamiento como el de transmisión.
(a) 90 tramas/seg.
(b) 900 tramas/seg.
(c) 9 000 tramas/seg.
2. Un grupo de *N* estaciones comparte un canal ALOHA puro de 56 kbps. La salida de cada estación es una trama de 1000 bits en promedio cada 100 seg, aun si la anterior no se ha enviado (por ejemplo, las estaciones pueden almacenar en búfer las tramas salientes). ¿Cuál es el valor máximo de *N*?
3. Considere el retardo del ALOHA puro comparándolo con el ALOHA ranurado cuando la carga es baja. ¿Cuál es menor? Explique su respuesta.
4. Una gran población de usuarios de ALOHA genera 50 solicitudes/seg, incluyendo tanto las originales como las retransmisiones. El tiempo se divide en ranuras de 40 mseg.
(a) ¿Cuál es la oportunidad de éxito en el primer intento?
(b) ¿Cuál es la probabilidad de que haya exactamente *k* colisiones y después un éxito?
(c) ¿Cuál es el número esperado de intentos de transmisión necesarios?
5. En un sistema ALOHA ranurado de población infinita, la cantidad promedio de número de ranuras que espera una estación entre una colisión y una retransmisión es de 4. Grafique; la curva de retardo contra velocidad real de transmisión para este sistema.
6. ¿Cuál es la longitud de una ranura de contención en CSMA/CD para (a) un cable con dos conductores de 2 km (la velocidad de propagación de la señal es 82% de la velocidad de propagación de la señal en vacío) y, (b) un cable de fibra óptica multimodo de 40 km (la velocidad de propagación de la señal es 65% de la velocidad de propagación de la señal en vacío)?
7. ¿Cuánto debe esperar una estación, *s*, en el peor de los casos, antes de empezar a transmitir su trama sobre una LAN que utiliza el protocolo básico de mapa de bits?
8. En el protocolo de conteo binario descendente, explique cómo se puede prohibir a una estación de menor numeración que envíe un paquete.
9. Dieciséis estaciones, numeradas del 1 al 16, contienden por el uso de un canal compartido mediante el protocolo de recorrido de árbol adaptable. Si todas las estaciones cuyas direcciones son números primos de pronto quedaran listas al mismo tiempo, ¿cuántas ranuras de bits se necesitan para resolver la contención?
10. Considere cinco estaciones inalámbricas, *A*, *B*, *C*, *D* y *E*. La estación *A* se puede comunicar con todas las demás. *B* se puede comunicar con *A*, *C* y *E*. *C* se puede comunicar con *A*, *B* y *D*. *D* se puede comunicar con *A*, *C* y *E*. *E* se puede comunicar con *A*, *D* y *B*.
(a) Cuando *A* envía a *B*, ¿qué otras comunicaciones son posibles?
(b) Cuando *B* envía a *A*, ¿qué otras comunicaciones son posibles?
(c) Cuando *B* envía a *C*, ¿qué otras comunicaciones son posibles?
11. Seis estaciones, de *A* a *F*, se comunican mediante el protocolo MACA. ¿Es posible que dos transmisiones tengan lugar de manera simultánea? Explique su respuesta.

12. Un edificio de oficinas de siete pisos tiene 15 oficinas adyacentes por piso. Cada oficina contiene un enchufe de pared para una terminal en la pared frontal, por lo que los enchufes forman una rejilla rectangular en el plano vertical, con una separación de 4 m entre enchufes, tanto en forma vertical como horizontal. Suponiendo que es factible tender un cable recto entre cualquier par de enchufes, en forma horizontal, vertical o diagonal, ¿cuántos metros de cable se necesitan para conectar todos los enchufes usando
(a) una configuración en estrella con un solo enrutador en medio?
(b) una LAN 802.3 clásica?

13. ¿Cuál es la tasa de baudios de la Ethernet de 10 Mbps clásica?

14. Bosqueje la codificación Manchester en una Ethernet clásica para el flujo de bits: 0001110101.

15. Una LAN CSMA/CD (no la 802.3) de 10 Mbps y 1 km de largo tiene una velocidad de propagación de 200 m/μseg. En este sistema no se permiten los repetidores. Las tramas de datos tienen 256 bits de longitud, incluidos 32 bits de encabezado, suma de verificación y otra sobrecarga. La primera ranura de bits tras una transmisión exitosa se reserva para que el receptor capture el canal y envíe una trama de confirmación de recepción de 32 bits. ¿Cuál es la tasa de datos efectiva, excluyendo la sobrecarga, suponiendo que no hay colisiones?

16. Dos estaciones CSMA/CD intentan transmitir archivos grandes (multitrama). Después de enviar cada trama, contienden por el canal usando el algoritmo de retroceso exponencial binario. ¿Cuál es la probabilidad de que la contención termine en la ronda *k*, y cuál es el número promedio de rondas por periodo de contención?

17. Un paquete IP que se transmitirá a través de Ethernet tiene 60 bytes de longitud, incluyendo todos los encabezados. Si no se utiliza LLC, ¿se necesita relleno en la trama Ethernet, y de ser así, cuántos bytes?

18. Las tramas Ethernet deben tener al menos 64 bytes de longitud para asegurar que el transmisor permanezca en línea en caso de que ocurra una colisión en el extremo más lejano del cable. Fast Ethernet tiene el mismo tamaño mínimo de trama de 64 bytes pero puede recibir los bits diez veces más rápido. ¿Cómo es posible mantener el mismo tamaño mínimo de trama?

19. Algunos libros citan que el tamaño máximo de una trama Ethernet es de 1522 bytes en lugar de 1500. ¿Están en un error? Explique su respuesta.

20. ¿Cuántas tramas por segundo puede manejar Gigabit Ethernet? Reflexione con cuidado y tome en cuenta todos los casos relevantes. *Sugerencia*: es importante el hecho de que se trata de *Gigabit* Ethernet.

21. Mencione dos redes que permitan empaquetar tramas una tras otra. ¿Por qué es importante esta característica?

22. En la figura 4-27 se muestran cuatro estaciones, *A*, *B*, *C* y *D*. ¿Cuál de las dos últimas estaciones cree usted que está más cerca de *A* y por qué?

23. Proporcione un ejemplo para mostrar que el método RTS/CTS en el protocolo 802.11 es un poco distinto del protocolo MACA.

24. Una red LAN inalámbrica con un AP tiene 10 estaciones cliente. Cuatro estaciones tienen tasas de datos de 6 Mbps, cuatro tienen tasas de datos de 18 Mbps y las últimas dos tienen tasas de datos de 54 Mbps. ¿Cuál es la tasa de datos experimentada por cada estación cuando las 10 están enviando datos al mismo tiempo, y
(a) No se usa TXOP?
(b) Se usa TXOP?

25. Suponga que una LAN 802.11b de 11 Mbps transmite tramas de 64 bytes una tras otra, a través de un canal de radio con una tasa de error de bits de 10^{-7}. ¿Cuántas tramas por segundo resultarán dañadas en promedio?

26. Una red 802.16 tiene un ancho de canal de 20 MHz. ¿Cuántos bits/seg se pueden enviar a una estación suscrita?

27. Dé dos razones por las cuales las redes podrían usar un código de corrección de errores en lugar de detección de errores y retransmisión.

28. Mencione dos formas en las que WiMAX sea similar a 802.11, y dos formas en las que sea diferente de 802.11.

29. En la figura 4-34 podemos ver que un dispositivo Bluetooth puede estar en 2 piconets al mismo tiempo. ¿Hay alguna razón por la cual un dispositivo no pueda fungir como maestro en ambas al mismo tiempo?

30. ¿Cuál es el tamaño máximo del campo de datos para una trama Bluetooth de tres ranuras, con la tasa de transmisión básica? Explique su respuesta.

31. La figura 4-24 muestra varios protocolos de capa física. ¿Cuál de éstos está más cercano al protocolo de capa física Bluetooth? ¿Cuál es la principal diferencia entre ambos?

32. En la sección 4.6.6 se menciona que la eficiencia de una trama de 1 ranura con codificación de repetición es de al rededor de 13% con la tasa de datos básica. ¿Cuál será la eficiencia si ahora se usa una trama de cinco ranuras con codificación de repetición y una tasa de datos básica?

33. Las tramas baliza (beacon) en el espectro disperso con salto de frecuencia, variante del 802.11, contienen el tiempo de permanencia. ¿Cree usted que las tramas baliza análogas de Bluetooth también contienen tiempo de permanencia? Explique su respuesta.

34. Suponga que hay 10 etiquetas RFID alrededor de un lector RFID. ¿Cuál es el mejor valor de Q? ¿Qué tan probable es que una etiqueta responda sin colisión en una ranura dada?

35. Mencione algunos de los aspectos de seguridad de un sistema RFID.

36. Un switch diseñado para usarse con Fast Ethernet tiene un plano posterior que puede transportar 10 Gbps. ¿Cuántas tramas/seg puede manejar en el peor caso?

37. Describa brevemente la diferencia entre los switches de almacenamiento y reenvío, y los switches de conmutación al vuelo (*cut-through*).

38. Considere la LAN extendida que se conecta mediante el uso de los puentes *B1* y *B2* en la figura 4-41(b). Suponga que las tablas de hash en los dos puentes están vacías. Mencione todos los puertos en los que se reenviará un paquete para la siguiente secuencia de transmisiones de datos:
(a) *A* envía un paquete a *C*.
(b) *E* envía un paquete a *F*.
(c) *F* envía un paquete a *E*.
(d) *G* envía un paquete a *E*.
(e) *D* envía un paquete a *A*.
(f) *B* envía un paquete a *F*.

39. Los conmutadores de almacenamiento y reenvío tienen una ventaja sobre los de conmutación al vuelo (*cut-through*) respecto a las tramas dañadas. Explique cuál es.

40. En la sección 4.8.3 se menciona que algunos puentes pueden incluso no estar presentes en el árbol de expansión. Describa un escenario en el que un puente pueda no estar presente en el árbol de expansión.

41. Para que las redes VLAN funcionen, se necesitan tablas de configuración en los puentes. ¿Qué pasaría si las VLAN de la figura 4-47 utilizaran hubs en vez de switches? ¿Los hubs también necesitarían tablas de configuración? ¿Por qué sí o por qué no?

42. En la figura 4-48, el switch en el dominio final heredado en la parte derecha tiene soporte para VLAN. ¿Sería posible utilizar ahí un switch heredado? Si es así, ¿cómo funcionaría? En caso contrario, ¿por qué no?

43. Escriba un programa para simular el comportamiento del protocolo CSMA/CD sobre Ethernet cuando hay *N* estaciones listas para transmitir en el momento en que se está transmitiendo una trama. Su programa deberá informar las veces que cada estación inicia con éxito el envío de su trama. Suponga que un pulso de reloj ocurre una vez por cada ranura de tiempo (51.2 μseg) y que la detección de una colisión y el envío de una secuencia atorada tarda una ranura de tiempo. Todas las tramas tienen la longitud máxima permitida.

5

LA CAPA DE RED

La capa de red se encarga de llevar los paquetes todo el camino, desde el origen hasta el destino. Para llegar al destino tal vez sea necesario realizar muchos saltos en el camino por enrutadores intermedios. Esta función ciertamente contrasta con la de la capa de enlace de datos, cuya única meta es mover tramas de un extremo del cable al otro. Por lo tanto, la capa de red es la capa más baja que maneja la transmisión de extremo a extremo.

Para lograr sus objetivos, la capa de red debe conocer la topología de la red (es decir, el conjunto de todos los enrutadores y enlaces) y elegir las rutas apropiadas incluso para redes más grandes. También debe tener cuidado al escoger las rutas para no sobrecargar algunas de las líneas de comunicación y los enrutadores, y dejar inactivos a otros. Por último, cuando el origen y el destino están en redes diferentes, ocurren nuevos problemas. La capa de red es la encargada de solucionarlos. En este capítulo estudiaremos todos estos temas y los ilustraremos, principalmente mediante el uso de Internet y su protocolo de capa de red, IP.

5.1 ASPECTOS DE DISEÑO DE LA CAPA DE RED

En las siguientes secciones presentaremos una introducción sobre algunos de los problemas a los que se deben enfrentar los diseñadores de la capa de red. Estos temas incluyen el servicio proporcionado a la capa de transporte y el diseño interno de la red.

5.1.1 Conmutación de paquetes de almacenamiento y reenvío

Antes de empezar a explicar los detalles sobre la capa de red, vale la pena volver a exponer el contexto en el que operan los protocolos de esta capa. En la figura 5-1 podemos ver este con-

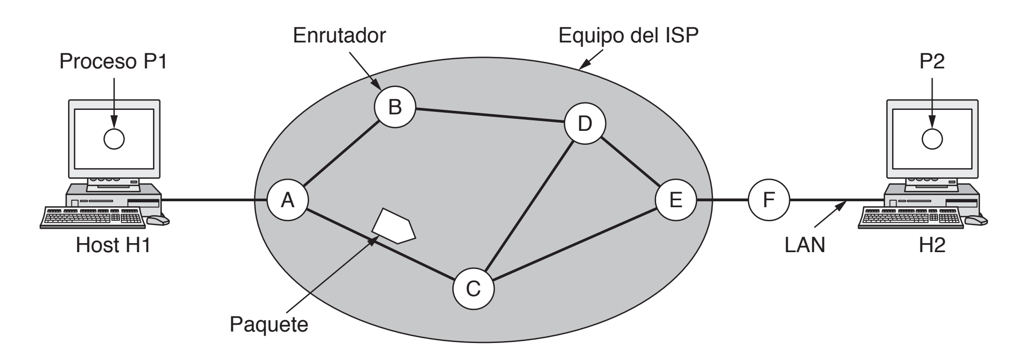


Figura 5-1. El entorno de los protocolos de la capa de red.

texto. Los componentes principales de la red son el equipo del Provedor del Servicio de Internet (ISP) (enrutadores conectados mediante líneas de transmisión), que se muestra dentro del óvalo sombreado, y el equipo de los clientes, que se muestra fuera del óvalo. El host *H1* está conectado de manera directa a un enrutador del ISP, *A*, tal vez en forma de una computadora en el hogar conectada a un módem DSL. En contraste, *H2* se encuentra en una LAN (que podría ser una Ethernet de oficina) con un enrutador, *F*, el cual es propiedad del cliente, quien lo maneja. Este enrutador tiene una línea alquilada que va al equipo del ISP. Mostramos a *F* fuera del óvalo porque no pertenece al ISP. Sin embargo y para los fines de este capítulo, los enrutadores locales de los clientes se consideran parte de la red del ISP debido a que ejecutan los mismos algoritmos que los enrutadores del ISP (y nuestro principal interés aquí son los algoritmos).

Este equipo se utiliza de la siguiente manera. Un host que desea enviar un paquete lo transmite al enrutador más cercano, ya sea en su propia LAN o a través de un enlace punto a punto que va al ISP. El paquete se almacena ahí hasta que haya llegado por completo y el enlace haya terminado su procesamiento mediante la comprobación de la suma de verificación. Después se reenvía al siguiente enrutador de la ruta hasta que llega al host de destino, en donde se entrega. Este mecanismo se denomina conmutación de almacenamiento y envío, como hemos visto en capítulos anteriores.

5.1.2 Servicios proporcionados a la capa de transporte

La capa de red proporciona servicios a la capa de transporte en la interfaz entre la capa de red y de transporte. Una pregunta importante es qué tipo de servicios proporciona precisamente la capa de red a la capa de transporte. Hay que diseñar los servicios de manera cuidadosa, con los siguientes objetivos en mente:

1. Los servicios deben ser independientes de la tecnología del enrutador.
2. La capa de transporte debe estar aislada de la cantidad, tipo y topología de los enrutadores presentes.
3. Las direcciones de red disponibles para la capa de transporte deben usar un plan de numeración uniforme, incluso a través de redes LAN y WAN.

Dadas estas metas, los diseñadores de la capa de red tienen mucha libertad para escribir especificaciones detalladas de los servicios que se ofrecerán a la capa de transporte. Con frecuencia esta libertad degenera en una batalla campal entre dos bandos en conflicto. La discusión se centra en determinar si la capa de red debe proporcionar un servicio orientado a conexión o un servicio sin conexión.

Un bando (representado por la comunidad de Internet) declara que la tarea de los enrutadores es mover paquetes de un lado a otro, y nada más. Desde su punto de vista (basado en casi 40 años de experiencia con una red de computadoras real), la red es de naturaleza no confiable, sin importar su diseño. Por lo tanto, los hosts deben aceptar este hecho y efectuar ellos mismos el control de errores (es decir, detección y corrección de errores) y el control de flujo.

Este punto de vista conduce a la conclusión de que el servicio de red debe ser sin conexión y debe contar tan sólo con las primitivas SEND PACKET y RECEIVE PACKET. En particular, no debe efectuarse ningún ordenamiento de paquetes ni control de flujo, pues de todos modos los hosts lo van a efectuar y por lo general se obtiene poca ganancia al hacerlo dos veces. Este razonamiento es un ejemplo del **argumento extremo a extremo** (*end-to-end argument*), un principio de diseño que ha sido muy influyente para dar forma a Internet. (Saltzer y colaboradores, 1984). Además, cada paquete debe llevar la dirección de destino completa, porque cada paquete enviado se transporta de manera independiente a sus antecesores, si los hay.

El otro bando (representado por las compañías telefónicas) argumenta que la red debe proporcionar un servicio confiable, orientado a conexión. Afirman que los 100 años de experiencia exitosa con el sistema telefónico mundial, son una excelente guía. Desde este punto de vista, la calidad del servicio es el factor dominante y, sin conexiones en la red, tal calidad es muy difícil de alcanzar, en especial para el tráfico de tiempo real como la voz y el video.

Incluso después de varias décadas, esta controversia aún sigue muy vigente. Las primeras redes de datos que se utilizaron ampliamente, como X.25 en la década de 1970 y su sucesora, Frame Relay en la década de 1980, eran orientadas a conexión. Sin embargo, desde los días de la ARPANET y la Internet en sus inicios, la popularidad de las capas de red sin conexión ha aumentado en forma considerable. Ahora el protocolo IP es un símbolo constante de éxito. No se vio afectado por una tecnología orientada a conexión llamada ATM, que se desarrolló para desbancarlo en la década de 1980; en cambio, ahora ATM se usa en nichos y es IP quien domina las redes telefónicas. Sin embargo, tras bambalinas Internet desarrolla características orientadas a conexión a medida que la calidad del servicio adquiere más importancia. Dos ejemplos de tecnologías orientadas a conexión son MPLS (Conmutación Multiprotocolo Mediante Etiquetas), que describiremos en este capítulo, y las redes VLAN que vimos en el capítulo 4. Ambas tecnologías se utilizan ampliamente.

5.1.3 Implementación del servicio sin conexión

Puesto que ya vimos las dos clases de servicios que la capa de red puede proporcionar a sus usuarios, es tiempo de analizar el funcionamiento interno de esta capa. Se pueden realizar dos formas de organización distintas, dependiendo del tipo de servicio ofrecido. Si se ofrece el servicio sin conexión, los paquetes se transmiten por separado en la red y se enrutan de manera independiente. No se necesita una configuración por adelantado. En este contexto, por lo general los paquetes se conocen como **datagramas** (en analogía con los telegramas) y la red se conoce como **red de datagramas**. Si se utiliza el servicio orientado a conexión, hay que establecer una ruta del enrutador de origen al enrutador de destino antes de poder enviar cualquier paquete de datos. Esta conexión se conoce como **VC** (**circuito virtual**), en analogía con los circuitos físicos establecidos por el sistema telefónico, y la red se denomina **red de circuitos virtuales**. En esta sección examinaremos las redes de datagramas; en la siguiente analizaremos las redes de circuitos virtuales.

Ahora veamos cómo funciona una red de datagramas. Suponga que el proceso *P1* de la figura 5-2 tiene un mensaje largo para *P2*. Dicho proceso entrega el mensaje a la capa de transporte y le indica a ésta que lo envíe al proceso *P2* en el host *H2*. El código de la capa de transporte se ejecuta en *H1*, por lo general dentro del sistema operativo. Dicho código agrega un encabezado de transporte al frente del mensaje y entrega el resultado a la capa de red, que quizá sólo sea otro procedimiento dentro del sistema operativo.

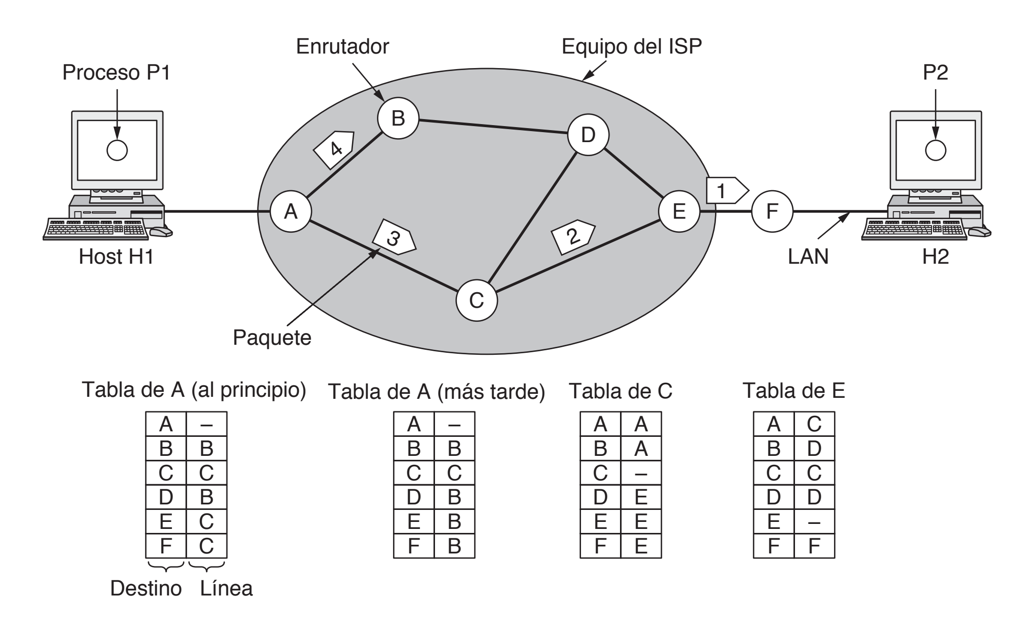


Tabla de A (al principio)

Destino	Línea
A	–
B	B
C	C
D	B
E	C
F	C

Tabla de A (más tarde)

A	–
B	B
C	C
D	B
E	B
F	B

Tabla de C

A	A
B	A
C	–
D	E
E	E
F	E

Tabla de E

A	C
B	D
C	C
D	D
E	–
F	F

Figura 5-2. Enrutamiento dentro de una red de datagramas.

Supongamos para este ejemplo que el mensaje es cuatro veces más largo que el tamaño máximo de paquete, por lo que la capa de red tiene que dividirlo en cuatro paquetes: 1, 2, 3 y 4; y enviar cada uno por turnos al enrutador *A* mediante algún protocolo punto a punto; por ejemplo, PPP. En este momento entra en acción el ISP. Cada enrutador tiene una tabla interna que le indica a dónde enviar paquetes para cada uno de los posibles destinos. Cada entrada en la tabla es un par que consiste en un destino y la línea de salida que se utilizará para ese destino. Sólo se pueden utilizar líneas conectadas en forma directa. Por ejemplo, en la figura 5-2, *A* sólo tiene dos líneas de salida (a *B* y a *C*), por lo que cada paquete entrante se debe enviar a uno de estos enrutadores, incluso si el destino final es algún otro enrutador. En la figura 5-2, la tabla de enrutamiento inicial de *A* se muestra bajo la leyenda “al principio”.

En *A*, los paquetes 1, 2 y 3 se almacenan unos momentos, después de haber llegado por el enlace entrante y de haber comprobado sus sumas de verificación. Después cada paquete se reenvía de acuerdo con la tabla de *A*, por el enlace de salida a *C* dentro de una nueva trama. Después, el paquete 1 se reenvía a *E* y después a *F*. Cuando llega a *F*, se envía dentro de una trama a *H2* a través de la LAN. Los paquetes 2 y 3 siguen la misma ruta.

Sin embargo, ocurre algo diferente con el paquete 4. Cuando llega a *A* se envía al enrutador *B*, aun cuando también está destinado a *F*. Por alguna razón, *A* decidió enviar el paquete 4 por una ruta diferente a la de los primeros tres paquetes. Tal vez se enteró de que había alguna congestión de tráfico en alguna parte de la ruta *ACE* y actualizó su tabla de enrutamiento, como se muestra bajo la leyenda “más tarde”. El algoritmo que maneja las tablas y realiza las decisiones de enrutamiento se conoce como **algoritmo de enrutamiento**. Los algoritmos de enrutamiento son uno de los principales temas que estudiaremos en este capítulo. Hay distintos tipos de ellos, como veremos más adelante.

IP (Protocolo Internet), que constituye la base de Internet, es el ejemplo dominante de un servicio de red sin conexión. Cada paquete transporta una dirección IP de destino que los enrutadores usan para reenviar cada paquete por separado. Las direcciones son de 32 bits en los paquetes IPv4 y de 128 bits en los paquetes IPv6. Más adelante en este capítulo describiremos IP con mucho más detalle.

5.1.4 Implementación del servicio orientado a conexión

Para el servicio orientado a conexión necesitamos una red de circuitos virtuales. Veamos ahora cómo funciona. La idea detrás de los circuitos virtuales es evitar la necesidad de elegir una nueva ruta para cada paquete enviado, como en la figura 5-2. En cambio, cuando se establece una conexión, se elige una ruta de la máquina de origen a la máquina de destino como parte de la configuración de conexión y se almacena en tablas dentro de los enrutadores. Esa ruta se utiliza para todo el tráfico que fluye a través de la conexión, de la misma forma en que funciona el sistema telefónico. Cuando se libera la conexión, también se termina el circuito virtual. Con el servicio orientado a conexión, cada paquete lleva un identificador que indica a cuál circuito virtual pertenece.

Como ejemplo, considere la situación que se muestra en la figura 5-3. En ésta, el host *H1* ha establecido una conexión 1 con el host *H2*. Esta conexión se recuerda como la primera entrada en cada una de las tablas de enrutamiento. La primera línea de la tabla *A* indica que si un paquete con el identificador de conexión 1 viene de *H1*, se enviará al enrutador *C* y se le dará el identificador de conexión 1. De manera similar, la primera entrada en *C* enruta el paquete a *E*, también con el identificador de conexión 1.

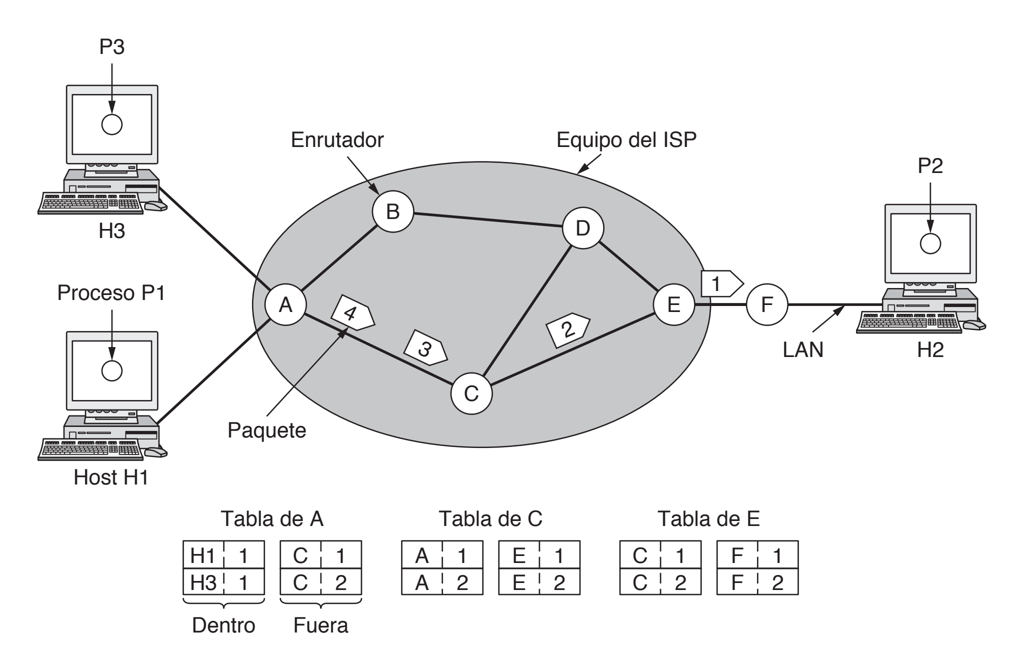


Figura 5-3. Enrutamiento dentro de una red de circuitos virtuales.

Ahora consideremos lo que sucede si *H3* también desea establecer una conexión con *H2*. Elige el identificador de conexión 1 (debido a que está iniciando la conexión y a que ésta es su única conexión) y le indica a la red que establezca el circuito virtual. Esto nos lleva a la segunda fila de las tablas. Observe que aquí surge un problema, pues aunque *A* sí puede saber con facilidad cuáles paquetes de conexión 1 provienen de *H1* y cuáles provienen de *H3*, *C* no puede hacerlo. Por esta razón, *A* asigna un identificador de conexión diferente al tráfico de salida para la segunda conexión. Evitar conflictos de este tipo es la razón por la cual los enrutadores necesitan la habilidad de reemplazar identificadores de conexión en los paquetes de salida.

En algunos contextos, a este proceso se le conoce como **conmutación mediante etiquetas**. **MPLS** (**Conmutación Multiprotocolo Mediante Etiquetas**, del inglés *MultiProtocol Label Swithching*) es un

ejemplo de servicio de red orientado a conexión. Se utiliza dentro de las redes de ISP en Internet, en donde los paquetes IP se envuelven en un encabezado MPLS que tiene un identificador de conexión o etiqueta de 20 bits. A menudo el servicio MPLS se oculta de los clientes y el ISP establece conexiones de largo plazo para grandes cantidades de tráfico, pero cada vez se utiliza más para ayudar cuando la calidad del servicio es importante, al igual que para otras tareas de administración de tráfico de ISP. Más adelante en este capítulo veremos más detalles sobre MPLS.

5.1.5 Comparación entre las redes de circuitos virtuales y las redes de datagramas

Tanto los circuitos virtuales como los datagramas tienen sus seguidores y sus detractores. Ahora intentaremos resumir los argumentos de ambos bandos. Los aspectos principales se listan en la figura 5-4, aunque es probable que los puristas puedan encontrar ejemplos contrarios para todo lo indicado en la figura.

Asunto	Red de datagramas	Red de circuitos virtuales
Configuración del circuito.	No necesaria.	Requerida.
Direccionamiento.	Cada paquete contiene la dirección de origen y de destino completas.	Cada paquete contiene un número de CV corto.
Información de estado.	Los enrutadores no contienen información de estado sobre las conexiones.	Cada CV requiere espacio de tabla del enrutador por cada conexión.
Enrutamiento.	Cada paquete se enruta de manera independiente.	La ruta se elije cuando se establece el CV; todos los paquetes siguen esa ruta.
Efecto de fallas del enrutador.	Ninguno, excepto para paquetes perdidos durante una caída.	Terminan todos los CVs que pasaron por el enrutador defectuoso.
Calidad del servicio.	Difícil.	Fácil si se pueden asignar suficientes recursos por adelantado para cada CV.
Control de congestión.	Difícil.	Fácil si se pueden asignar suficientes recursos por adelantado para cada CV.

Figura 5-4. Comparación entre las redes de datagramas y de circuitos virtuales.

Dentro de la red existen ventajas y desventajas entre los circuitos virtuales y los datagramas. Una de ellas tiene que ver con el tiempo de configuración y el tiempo de análisis de la dirección. El uso de circuitos virtuales requiere una fase de configuración que necesita tiempo y recursos. Sin embargo, una vez que se paga este precio, es fácil averiguar qué hacer con un paquete de datos en una red de circuitos virtuales: el enrutador sólo usa el número de circuito para buscar en una tabla y encontrar hacia dónde va el paquete. En una red de datagramas no se requiere configuración, pero se requiere un procedimiento más complicado para localizar la entrada correspondiente al destino.

Un aspecto relacionado es que las direcciones de destino que se utilizan en las redes de datagramas son más largas que los números de los circuitos que se utilizan en las redes de circuitos virtuales, ya que tienen un significado global. Si los paquetes tienden a ser bastante cortos, incluir una dirección de destino completa en cada paquete puede representar una cantidad considerable de sobrecarga y, por ende, un desperdicio de ancho de banda.

Otro aspecto más es la cantidad de espacio de tabla requerido en la memoria del enrutador. Una red de datagramas necesita tener una entrada para cada destino posible, mientras que una red de circui-

tos virtuales sólo necesita una entrada para cada circuito virtual. Sin embargo, esta ventaja es engañosa debido a que los paquetes de configuración de conexión también tienen que enrutarse y utilizan direcciones de destino, al igual que los datagramas.

Los circuitos virtuales tienen ciertas ventajas en cuanto a garantizar la calidad del servicio y evitar congestiones en la red, pues los recursos (por ejemplo, búferes, ancho de banda y ciclos de CPU) se pueden reservar por adelantado al establecer la conexión. Una vez que comienzan a llegar los paquetes, el ancho de banda y la capacidad de enrutamiento necesarios estarán ya disponibles. En una red de datagramas es más difícil evitar la congestión.

En los sistemas de procesamiento de transacciones (por ejemplo, las tiendas que llaman para verificar compras con tarjeta de crédito), la sobrecarga requerida para establecer y terminar un circuito virtual puede ocupar mucho más tiempo que el uso real del circuito. Si la mayor parte del tráfico esperado es de este tipo, el uso de circuitos virtuales dentro de la red tiene poco sentido. Por otra parte, para usos en donde el tiempo de ejecución sea extenso, como el tráfico VPN entre dos oficinas corporativas, pueden ser de utilidad los circuitos virtuales permanentes (que se establecen en forma manual y duran meses o años).

Los circuitos virtuales también tienen un problema de vulnerabilidad. Si falla un enrutador y pierde su memoria, incluso aunque vuelva a funcionar un segundo después, todos los circuitos virtuales que pasan por él tendrán que abortarse. En contraste, si un enrutador de datagramas falla, sólo sufrirán los usuarios cuyos paquetes se pusieron en cola en el enrutador en ese momento (y tal vez ni siquiera ellos, ya que es probable que el emisor los retransmita poco tiempo después). La pérdida de una línea de comunicación es fatal para los circuitos virtuales que la usan, pero se puede compensar con facilidad si se usan datagramas. Éstos también permiten que los enrutadores balanceen el tráfico a través de la red, ya que las rutas se pueden cambiar a lo largo de una secuencia extensa de transmisiones de paquetes.

5.2 ALGORITMOS DE ENRUTAMIENTO

La principal función de la capa de red es enrutar paquetes de la máquina de origen a la de destino. En la mayoría de las redes, los paquetes requerirán varios saltos para completar el viaje. La única excepción importante son las redes de difusión, pero aun aquí es importante el enrutamiento si el origen y el destino no están en el mismo segmento de red. Los algoritmos que eligen las rutas y las estructuras de datos que usan constituyen un aspecto principal del diseño de la capa de red.

El **algoritmo de enrutamiento** es aquella parte del software de la capa de red responsable de decidir por cuál línea de salida se transmitirá un paquete entrante. Si la red usa datagramas de manera interna, esta decisión debe tomarse cada vez que llega un paquete de datos, dado que la mejor ruta podría haber cambiado desde la última vez. Si la red usa circuitos virtuales internamente, las decisiones de enrutamiento se toman sólo al establecer un circuito virtual nuevo. En lo sucesivo, los paquetes de datos simplemente siguen la ruta ya establecida. Este último caso a veces se llama **enrutamiento de sesión**, dado que una ruta permanece vigente durante toda una sesión (por ejemplo, durante una sesión a través de una VPN).

En ocasiones es útil distinguir entre el enrutamiento, que es el proceso que consiste en tomar la decisión de cuáles rutas utilizar, y el reenvío, que consiste en la acción que se toma cuando llega un paquete. Podemos considerar que un enrutador tiene dos procesos internos. Uno de ellos maneja cada paquete conforme llega, y después busca en las tablas de enrutamiento la línea de salida por la cual se enviará. Este proceso se conoce como **reenvío**. El otro proceso es responsable de llenar y actualizar las tablas de enrutamiento. Es ahí donde entra en acción el algoritmo de enrutamiento.

Sin importar si las rutas se eligen de manera independiente para cada paquete enviado o sólo cuando se establecen nuevas conexiones, existen ciertas propiedades que todo algoritmo de enrutamiento debe

poseer: exactitud, sencillez, robustez, estabilidad, equidad y eficiencia. La exactitud y la sencillez apenas requieren comentarios, pero la necesidad de robustez puede ser menos obvia a primera vista. Una vez que una red principal entra en operación, es de esperar que funcione de manera continua durante años, sin fallas a nivel de sistema. Durante ese periodo habrá fallas de hardware y de software de todo tipo. Los hosts, enrutadores y líneas fallarán en forma repetida y la topología cambiará muchas veces. El algoritmo de enrutamiento debe ser capaz de manejar los cambios de topología y tráfico sin necesidad de abortar todas las tareas en todos los hosts. ¡Imagine el desastre si la red tuviera que reiniciarse cada vez que fallara un enrutador!

La estabilidad también es una meta importante para el algoritmo de enrutamiento. Existen algoritmos de enrutamiento que nunca convergen hacia un conjunto de rutas fijo, sin importar el tiempo que permanezcan en operación. Un algoritmo estable alcanza el equilibrio y lo conserva. Además debe converger con rapidez, ya que se puede interrumpir la comunicación hasta que el algoritmo de enrutamiento haya llegado a un equilibrio.

La equidad y la eficiencia pueden parecer obvias (sin duda, ninguna persona razonable se opondrá a ellas), pero resulta que con frecuencia son metas contradictorias. En la figura 5-5 se muestra un ejemplo sencillo de este conflicto. Suponga que hay suficiente tráfico entre A y A', entre B y B' y entre C y C' para saturar los enlaces horizontales. Con el fin de maximizar el flujo total, es necesario suspender el tráfico de X a X' por completo. Por desgracia, tal vez X y X' no lo vean de esa forma. Sin duda se requiere cierto compromiso entre la eficiencia global y la equidad hacia las conexiones individuales.

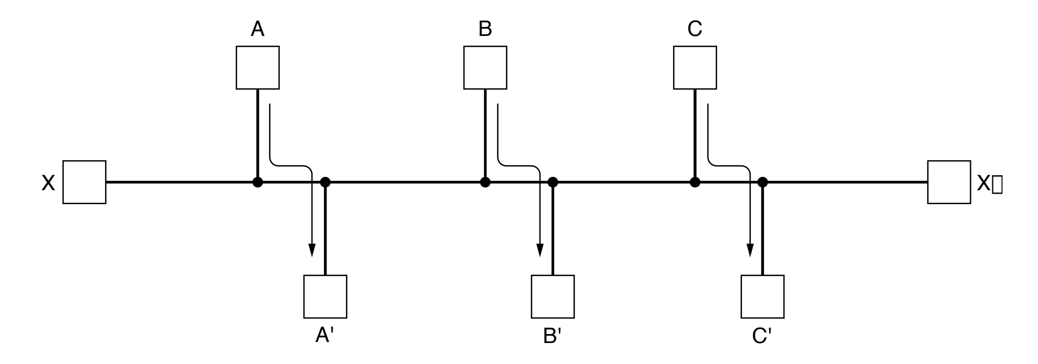


Figura 5-5. Una red con un conflicto entre equidad y eficiencia.

Antes de que podamos siquiera intentar encontrar el punto medio entre la equidad y la eficiencia, debemos decidir qué es lo que buscamos optimizar. Minimizar el retardo promedio de los paquetes es un candidato obvio para enviar tráfico a través de la red en forma efectiva, pero también lo es aumentar al máximo la velocidad real de transferencia total de la red. Además, estas dos metas también están en conflicto, ya que operar cualquier sistema de colas cerca de su capacidad máxima implica un gran retardo de encolamiento. Como término medio, muchas redes intentan minimizar la distancia que debe recorrer el paquete, o simplemente reducir el número de saltos que tiene que dar un paquete. Cualquiera de estas opciones tiende a mejorar el retardo y también reduce la cantidad de ancho de banda consumido por paquete, lo cual tiende a mejorar la velocidad de transferencia real de la red en general.

Podemos agrupar los algoritmos de enrutamiento en dos clases principales: no adaptativos y adaptativos. Los **algoritmos no adaptativos** no basan sus decisiones de enrutamiento en mediciones o estimaciones del tráfico y la topología actuales. En cambio, la decisión de qué ruta se usará para llegar de I a J (para todas las I y J) se calcula por adelantado, fuera de línea, y se descarga en los enrutadores al arrancar

la red. En ocasiones este procedimiento se denomina **enrutamiento estático**. Como no responde a las fallas, el enrutamiento estático es más útil para las situaciones en las que la elección de enrutamiento es clara. Por ejemplo, el enrutador *F* en la figura 5-3 debería enviar al enrutador *E* los paquetes que van dirigidos a la red, sin importar el destino final.

En contraste, los **algoritmos adaptativos** cambian sus decisiones de enrutamiento para reflejar los cambios de topología y algunas veces también los cambios en el tráfico. Estos algoritmos de **enrutamiento dinámico** difieren en cuanto al lugar de donde obtienen su información (por ejemplo, localmente, de los enrutadores adyacentes o de todos los enrutadores), el momento en que cambian sus rutas (por ejemplo, cuando cambia la topología o cada ΔT segundos, a medida que cambia la carga) y la métrica que se usa para la optimización (por ejemplo, distancia, número de saltos o tiempo estimado de tránsito).

En las siguientes secciones estudiaremos una variedad de algoritmos de enrutamiento. Estos algoritmos cubren otros modelos de distribución además de enviar un paquete de un origen a un destino. Algunas veces el objetivo es enviar el paquete a varios destinos, a todos los destinos o a un destino de entre varios pertenecientes a un conjunto. Todos los algoritmos de enrutamiento que describimos aquí toman decisiones con base en la topología; aplazaremos el tema de la posibilidad de las decisiones con base en los niveles de tráfico hasta la sección 5.3.

5.2.1 Principio de optimización

Antes de entrar en algoritmos específicos, puede ser útil señalar que es posible hacer una postulado general sobre las rutas óptimas sin importar la topología o el tráfico de la red. Este postulado se conoce como **principio de optimización** (Bellman, 1957) y establece que si el enrutador *J* está en la ruta óptima del enrutador *I* al enrutador *K*, entonces la ruta óptima de *J* a *K* también está en la misma ruta. Para ver esto, llamemos r_1 a la parte de la ruta de *I* a *J* y r_2 al resto de la ruta. Si existiera una ruta mejor que r_2 entre *J* y *K*, se podría concatenar con r_1 para mejorar la ruta de *I* a *K*, lo cual contradice nuestro postulado de que r_1r_2 es óptima.

Como consecuencia directa del principio de optimización, podemos ver que el grupo de rutas óptimas de todos los orígenes a un destino dado forman un árbol con raíz en el destino. Dicho árbol se conoce como **árbol sumidero** (o **árbol divergente**) y se ilustra en la figura 5-6(b), donde la métrica de distancia es el número de saltos. El objetivo de todos los algoritmos de enrutamiento es descubrir y usar los árboles sumidero para todos los enrutadores.

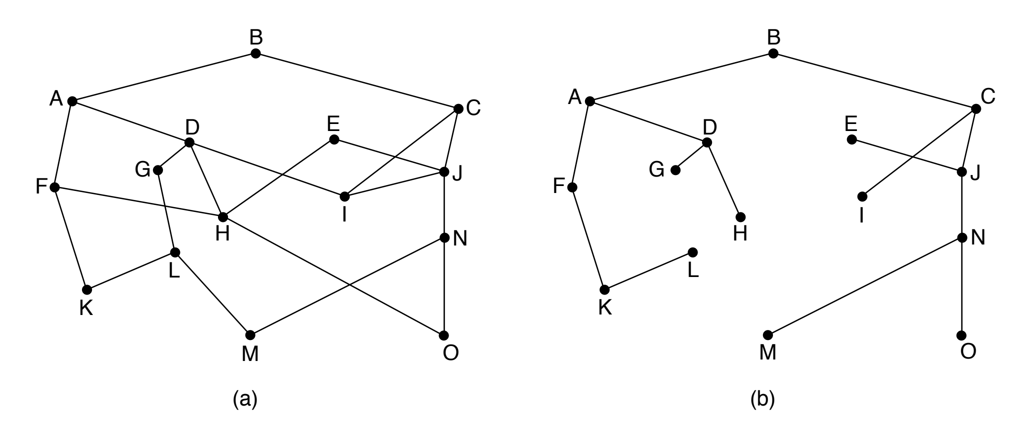


Figura 5-6. (a) Una red. (b) Un árbol sumidero para el enrutador *B*.

Cabe mencionar que un árbol sumidero no necesariamente es único; pueden existir otros árboles con las mismas longitudes de rutas. Si permitimos que se elijan todas las posibles rutas, el árbol se convierte en una estructura más general conocida como **DAG** (**Gráfico Acíclico Dirigido**, del inglés *Directed Acyclic Graph*). Los DAG no tienen ciclos. Usaremos los árboles sumidero como un método abreviado conveniente para ambos casos, que también dependen del supuesto técnico de que las rutas no interfieren entre sí; por ejemplo, un congestionamiento de tráfico en una ruta no provocará que se desvíe a otra ruta.

Puesto que un árbol sumidero ciertamente es un árbol, no contiene ciclos, por lo que cada paquete se entregará en un número de saltos finito y limitado. En la práctica, la vida no es tan fácil. Los enlaces y los enrutadores pueden fallar y recuperarse durante la operación, así que distintos enrutadores pueden tener ideas diferentes sobre la topología actual. Además, hemos evadido la cuestión de si cada enrutador tiene que adquirir de manera individual la información en la cual basa su cálculo del árbol sumidero, o si debe obtener esta información por otros medios. Regresaremos a estos asuntos pronto. Con todo, el principio de optimización y el árbol sumidero proporcionan los puntos de referencia contra los que se pueden medir otros algoritmos de enrutamiento.

5.2.2 Algoritmo de la ruta más corta

Empecemos nuestro estudio de los algoritmos de enrutamiento con una técnica simple para calcular las rutas óptimas con base en una imagen completa de la red. Estas rutas son las que queremos que encuentre un algoritmo de enrutamiento distribuido, aun cuando no todos los enrutadores conozcan todos los detalles de la red.

La idea es construir un grafo de la red, en donde cada nodo del grafo representa un enrutador y cada arco del grafo representa una línea o enlace de comunicaciones. Para elegir una ruta entre un par específico de enrutadores, el algoritmo simplemente encuentra la ruta más corta entre ellos en el grafo.

El concepto de la **ruta más corta** merece una explicación. Una manera de medir la longitud de una ruta es mediante el número de saltos. Con base en esta métrica, las rutas *ABC* y *ABE* en la figura 5-7 son igual de largas. Otra métrica es la distancia geográfica en kilómetros, en cuyo caso *ABC* es sin duda mucho más larga que *ABE* (suponiendo que la figura esté dibujada a escala).

Sin embargo, también son posibles muchas otras métricas además de los saltos y la distancia física. Por ejemplo, cada arco se podría etiquetar con el retardo promedio de un paquete de prueba estándar, determinado por una serie de pruebas cada hora. Con estas etiquetas en el grafo, la ruta más corta es la ruta más rápida, en lugar de la ruta con menos arcos o kilómetros.

En el caso general, las etiquetas de los arcos se podrían calcular como una función de la distancia, ancho de banda, tráfico promedio, costo de comunicación, retardo promedio y otros factores. Al cambiar la función de ponderación, el algoritmo calcularía la ruta "más corta" de acuerdo con cualquiera de estos criterios, o una combinación de ellos.

Se conocen varios algoritmos para calcular la ruta más corta entre dos nodos de un grafo. Uno de éstos se debe a Dijkstra (1959), el cual encuentra las rutas más cortas entre un origen y todos los destinos en una red. Cada nodo se etiqueta (entre paréntesis) con su distancia desde el nodo de origen a través de la mejor ruta conocida. Las distancias no deben ser negativas, como lo serán si se basan en cantidades reales como ancho de banda y retardo. Al principio no se conocen rutas, por lo que todos los nodos tienen la etiqueta infinito. A medida que avanza el algoritmo y se encuentran rutas, las etiquetas pueden cambiar para reflejar mejores rutas. Una etiqueta puede ser tentativa o permanente. En un principio todas las etiquetas son tentativas. Una vez que se descubre que una etiqueta representa la ruta más corta posible del origen a ese nodo, se vuelve permanente y no cambia más.

Para ilustrar el funcionamiento del algoritmo de etiquetado, observe el grafo ponderado no dirigido de la figura 5-7(a), donde las ponderaciones representan, por ejemplo, distancias. Queremos encontrar la ruta más

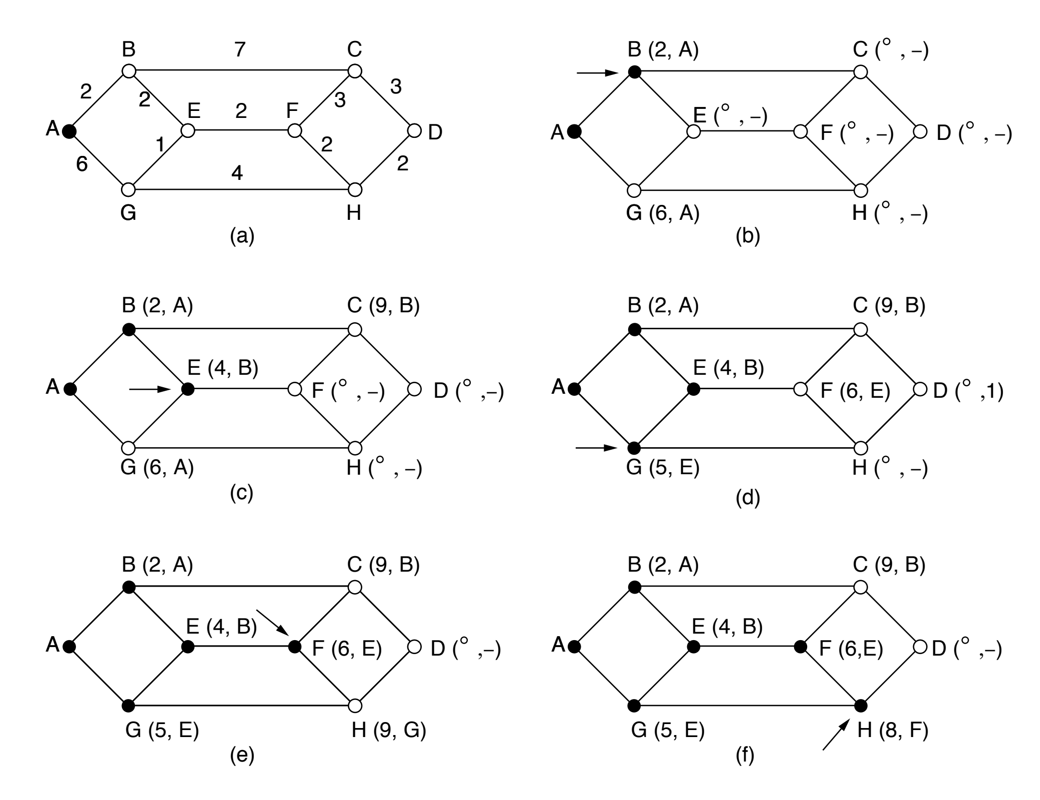


Figura 5-7. Los primeros seis pasos utilizados para calcular la ruta más corta de *A* a *D*. Las flechas indican el nodo de trabajo.

corta posible de *A* a *D*. Comenzamos por marcar el nodo *A* como permanente, lo cual se indica mediante un círculo relleno. Después examinamos, por turno, cada uno de los nodos adyacentes a *A* (el nodo de trabajo) y reetiquetamos cada uno de ellos con la distancia a *A*. Cada vez que reetiquetamos un nodo, también lo etiquetamos con el nodo desde el que se hizo la prueba, para poder reconstruir más tarde la ruta final. Si la red tuviera más de una ruta más corta de *A* a *D* y quisiéramos encontrarlas todas, tendríamos que recordar todos los nodos de prueba que podrían llegar a un nodo con la misma distancia.

Una vez que terminamos de examinar cada uno de los nodos adyacentes a *A*, examinamos todos los nodos etiquetados tentativamente en el grafo completo y hacemos permanente el de la etiqueta más pequeña, como se muestra en la figura 5-7(b). Éste se convierte en el nuevo nodo de trabajo.

Ahora comenzamos por *B* y examinamos todos los nodos adyacentes a él. Si la suma de la etiqueta en *B* y la distancia desde *B* hasta el nodo en consideración es menor que la etiqueta de ese nodo, tenemos una ruta más corta, por lo que reetiquetamos ese nodo.

Después de inspeccionar todos los nodos adyacentes al nodo de trabajo y cambiar las etiquetas tentativas si es posible, se busca en el grafo completo el nodo etiquetado tentativamente con el menor valor. Este nodo se hace permanente y se convierte en el nodo de trabajo para la siguiente ronda. En la figura 5-7 se muestran los primeros seis pasos del algoritmo.

Para ver por qué funciona el algoritmo, vea la figura 5-7(c). En ese punto acabamos de hacer permanente a *E*. Suponga que hubiera una ruta más corta que *ABE*, digamos *AXYZE* (para alguna *X* y *Y*). Hay dos posibilidades: el nodo Z ya se hizo o no permanente. Si ya es permanente, entonces *E* ya se probó (en la ronda que siguió a aquella en la que *Z* se hizo permanente), por lo que la ruta *AXYZE* no ha escapado de nuestra atención y, por lo tanto, no puede ser una ruta más corta.

Ahora considere el caso en el que *Z* aún está etiquetado de forma tentativa. Si la etiqueta en Z es mayor o igual que la de *E*, entonces *AXYZE* no puede ser una ruta más corta que *ABE*. Si la etiqueta es menor que la de *E*, entonces *Z* y no *E* se volverá permanente primero, lo que permitirá probar a *E* desde *Z*.

Este algoritmo se muestra en la figura 5-8. Las variables globales *n* y *dist* describen el grafo y se inicializan antes de llamar a *shortest_ path*. La única diferencia entre el programa y el algoritmo antes descrito es que, en la figura 5-8, calculamos la ruta más corta posible comenzando por el nodo terminal *t* en lugar del nodo de origen, *s*.

Debido a que las rutas más cortas desde *t* a *s* en un grafo no dirigido son iguales que las rutas más cortas de *s* a *t*, no importa en qué extremo empecemos. La razón de buscar hacia atrás es que cada nodo

```
#define MAX_NODES 1024                          /* número máximo de nodos */
#define INFINITY 1000000000                     /* un número mayor que cualquier ruta máxima */
int n, dist[MAX_NODES][MAX_NODES];              /* dist[i][ j] es la distancia de i a j */

void shortest_path(int s, int t, int path[])
{ struct state {                                /* la ruta en la que se está trabajando */
     int predecessor;                           /* nodo previo */
     int length;                                /* longitud del origen a este nodo */
     enum {permanent, tentative} label;         /* estado de la etiqueta */
  } state[MAX_NODES];

  int i, k, min;
  struct state *p;

  for (p = &state[0]; p < &state[n]; p++) {     /* estado de inicialización*/
     p->predecessor = -1;
     p->length = INFINITY;
     p->label = tentative;
  }
  state[t].length = 0; state[t].label = permanent;
  k = t;                                        /* k es el nodo de trabajo inicial */
  do{                                           /* ¿hay una ruta mejor desde k? */
     for (i = 0; i < n; i++)                    /* este grafo tiene n nodos */
          if (dist[k][i] != 0 && state[i].label == tentative) {
               if (state[k].length + dist[k][i] < state[i].length) {
                    state[i].predecessor = k;
                    state[i].length = state[k].length + dist[k][i];
               }
          }

     /* Encuentra el nodo etiquetado tentativamente con la etiqueta menor. */
     k = 0; min = INFINITY;
     for (i = 0; i < n; i++)
          if (state[i].label == tentative && state[i].length < min) {
               min = state[i].length;
               k = i;
          }
     state[k].label = permanent;
  } while (k != s);

  /* Copia la ruta en el arreglo de salida. */
  i = 0; k = s;
  do {path[i++] = k; k = state[k].predecessor; } while (k >= 0);
  }
```

Figura 5-8. Algoritmo de Dijkstra para calcular la ruta más corta a través de un grafo.

está etiquetado con su antecesor en vez de su sucesor. Al copiar la ruta final en la variable de salida, *path*, la ruta de salida se invierte. Al pasar esto, ambos efectos se cancelan y la respuesta se produce en el orden correcto.

5.2.3 Inundación

Cuando se implementa un algoritmo de enrutamiento, cada enrutador debe tomar decisiones con base en el conocimiento local, no en la imagen completa de la red. Una técnica local simple es la **inundación**, en la que cada paquete entrante se envía en todas las líneas de salida, excepto en la línea por la que llegó.

Sin duda la inundación genera grandes cantidades de paquetes duplicados; de hecho, puede ser una cantidad infinita a menos que se tomen algunas medidas para limitar el proceso. Una de estas medidas es integrar un contador de saltos al encabezado de cada paquete, que disminuya con cada salto, y que el paquete se descarte cuando el contador llegue a cero. Lo ideal es inicializar el contador de saltos con la longitud de la ruta entre el origen y el destino. Si el emisor desconoce el tamaño de la ruta, puede inicializar el contador para el peor caso; a saber, el diámetro total de la red.

La inundación con un conteo de saltos puede producir un número exponencial de paquetes duplicados a medida que aumenta el conteo de saltos y los enrutadores duplican los paquetes que ya han visto antes. Una técnica mejorada para ponerle diques a la inundación es llevar un registro de los paquetes difundidos en la inundación, para evitar enviarlos una segunda vez. Una manera de lograr este objetivo es hacer que el enrutador de origen ponga un número de secuencia en cada paquete que reciba de sus hosts. Así, cada enrutador necesita una lista por cada enrutador de origen que indique los números de secuencia originados en ese enrutador que ya ha visto. Si un paquete de entrada está en la lista, no se difunde mediante inundación.

Para evitar que la lista crezca sin límites, cada lista debe aumentar mediante un contador k que indique que ya se han visto todos los números de secuencia hasta k. Cuando llega un paquete, es fácil comprobar si éste ya se difundió mediante inundación (se compara su número de secuencia con k); de ser así, se descarta. Lo que es más, no se necesita la lista completa por debajo de k, ya que k la resume de manera efectiva.

La inundación no es práctica para enviar la mayoría de los paquetes, pero tiene algunos usos importantes. En primer lugar, asegura que un paquete se entregue en todos los nodos de la red. Esto podría ser un desperdicio si sólo hay un destino que necesite el paquete, pero es efectivo para difundir información. En las redes inalámbricas, todos los mensajes transmitidos por una estación los pueden recibir todas las demás estaciones dentro de su alcance de radio, lo cual de hecho se puede considerar como inundación; algunos algoritmos usan esta propiedad.

En segundo lugar, la inundación es en extremo robusta. Incluso si grandes cantidades de enrutadores vuelan en pedazos (por ejemplo, en una red militar ubicada en una zona de guerra), la inundación encontrará una ruta, si es que existe, para transmitir un paquete a su destino. Además, la inundación requiere muy poca configuración. Los enrutadores sólo necesitan conocer a sus vecinos. Esto significa que la inundación se puede usar como bloque de construcción para otros algoritmos de enrutamiento que son más eficientes pero requieren una configuración un poco más complicada. La inundación también se puede usar como métrica con la que se pueden comparar otros algoritmos de enrutamiento. La inundación siempre selecciona la ruta más corta debido a que selecciona todas las rutas posibles en paralelo. En consecuencia, ningún otro algoritmo puede producir un retardo más corto (si ignoramos la sobrecarga generada por el proceso de inundación mismo).

5.2.4 Enrutamiento por vector de distancia

Por lo general, las redes de computadoras utilizan algoritmos de enrutamiento dinámico más complejos que la inundación, pero más eficientes debido a que encuentran las rutas más cortas para la topología actual. En particular hay dos algoritmos dinámicos que son los más populares: el enrutamiento por vector de distancia y el enrutamiento por estado del enlace. En esta sección veremos el primer algoritmo. En la siguiente estudiaremos el segundo.

El algoritmo de **enrutamiento por vector de distancia** opera haciendo que cada enrutador mantenga una tabla (es decir, un vector) que proporcione la mejor distancia conocida a cada destino y el enlace que se puede usar para llegar ahí. Para actualizar estas tablas se intercambia información con los vecinos. Eventualmente, todos los ruteadores conocen el mejor enlace para alcanzar cada destino.

En ocasiones, el algoritmo de enrutamiento por vector de distancia recibe otros nombres, de los cuales el más común es algoritmo de enrutamiento **Bellman-Ford** distribuido, en honor a los investigadores que lo desarrollaron (Bellman, 1957; Ford y Fulkerson, 1962). Éste fue el algoritmo original de enrutamiento de ARPANET y también se usó en Internet con el nombre de RIP.

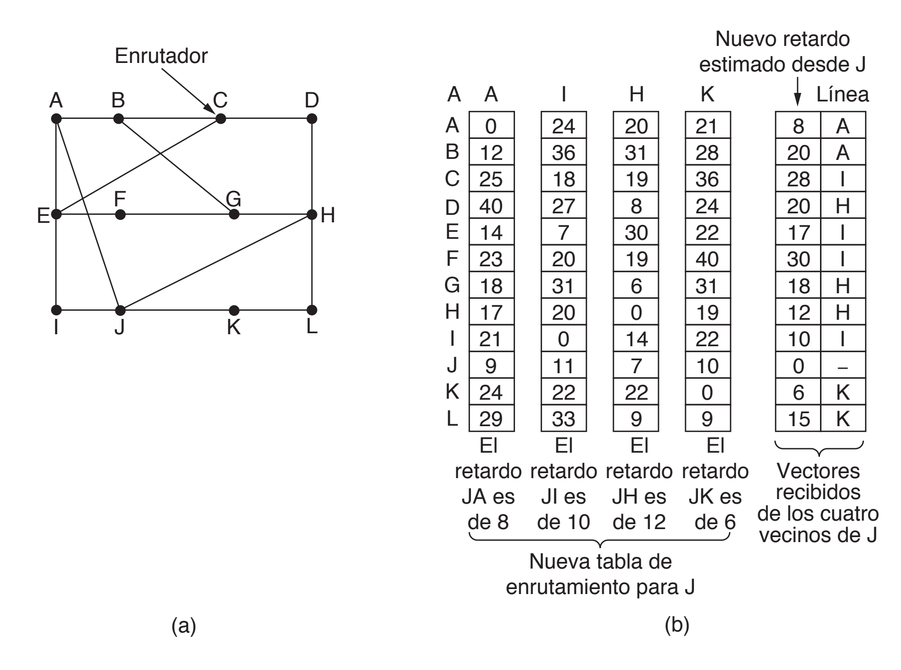
En el enrutamiento por vector de distancia, cada enrutador mantiene una tabla de enrutamiento indizada por (y que contiene una entrada de) cada enrutador de la red. Esta entrada consta de dos partes: la línea preferida de salida a usar para ese destino y una estimación del tiempo o distancia a ese destino. La distancia se podría medir como la cantidad de saltos, o se podría usar otra métrica, como vimos al calcular las rutas más cortas.

Se supone que el enrutador conoce la “distancia” a cada uno de sus vecinos. Si la métrica es de saltos, la distancia sólo es un salto. Si la métrica es el retardo de propagación, el enrutador puede medirlo en forma directa con paquetes especiales de ECO que el receptor sólo marca con la hora y lo regresa tan rápido como puede.

Por ejemplo, suponga que el retardo se usa como métrica y que el enrutador conoce el retardo a cada uno de sus vecinos. Una vez cada T mseg, cada enrutador envía a todos sus vecinos una lista de sus retardos estimados a cada destino. También recibe una lista similar de cada vecino. Imagine que una de estas tablas acaba de llegar del vecino X, en donde X_i es la estimación respecto al tiempo que le toma llegar al enrutador i. Si el enrutador sabe que el retardo a X es de m mseg, también sabe que puede llegar al enrutador i a través de X en $X_i + m$ mseg. Al efectuar este cálculo para cada vecino, un enrutador puede encontrar la estimación que parezca ser la mejor y usar esa estimación, así como el enlace correspondiente, en su nueva tabla de enrutamiento. Cabe mencionar que en este cálculo no se utiliza la antigua tabla de enrutamiento.

Este proceso de actualización se ilustra en la figura 5-9. En la parte (a) se muestra una red. Las primeras cuatro columnas de la parte (b) muestran los vectores de retardo recibidos de los vecinos del enrutador J. A indica que tiene un retardo de 12 mseg a B, un retardo de 25 mseg a C, un retardo de 40 mseg a D, etcétera. Suponga que J ha medido o estimado el retardo a sus vecinos A, I, H y K en 8, 10, 12 y 6 mseg, respectivamente.

Considere la manera en que J calcula su nueva ruta al enrutador G. Sabe que puede llegar a A en 8 mseg; además A indica ser capaz de llegar a G en 18 mseg, por lo que J sabe que puede contar con un retardo de 26 mseg a G si reenvía a A los paquetes destinados para G. Del mismo modo, calcula el retardo a G a través de I, H y K en 41 (31 + 10), 18 (6 + 12) y 37 (31 + 6) mseg, respectivamente. El mejor de estos valores es el 18, por lo que escribe una entrada en su tabla de enrutamiento para indicar que el retardo a G es de 18 mseg, y que la ruta que se utilizará es a través de H. Para los demás destinos se realiza el mismo cálculo; la nueva tabla de enrutamiento se muestra en la última columna de la figura.



A	A	I	H	K	Nuevo retardo estimado desde J	Línea
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	–
K	24	22	22	0	6	K
L	29	33	9	9	15	K

Figura 5-9. (a) Una red. (b) Entrada de *A*, *I*, *H*, *K* y la nueva tabla de enrutamiento para *J*.

El problema del conteo al infinito

Al proceso de establecer los enrutamientos con base en las mejores rutas a través de la red se le conoce como **convergencia**. El enrutamiento por vector de distancia es útil como una simple técnica mediante la cual los enrutadores pueden calcular las rutas más cortas en forma colectiva, pero tiene una grave desventaja en la práctica: aunque converge hacia la respuesta correcta, puede llegar a hacerlo con lentitud. Básicamente reacciona rápido a las buenas noticias, pero es lento con las malas. Considere un enrutador cuya mejor ruta al destino *X* es larga. Si en el siguiente intercambio el vecino *A* informa de manera repentina un retardo corto a *X*, el enrutador simplemente cambia y usa la línea *A* para enviar tráfico a *X*. En un intercambio de vectores, se procesan las buenas noticias.

Para ver la rapidez de propagación de las buenas noticias, considere la red de cinco nodos (lineal) de la figura 5-10, en donde la métrica de retardo es el número de saltos. Suponga que *A* está desactivado al principio y que los otros enrutadores lo saben. En otras palabras, todos registraron el retardo a *A* como infinito.

Al activarse *A*, los demás enrutadores saben de él gracias a los intercambios de vectores. Por sencillez, supondremos que hay un "gong" gigantesco en algún lado, golpeando periódicamente para iniciar de manera simultánea un intercambio de vectores entre todos los enrutadores. En el momento del primer intercambio, *B* se entera de que su vecino de la izquierda tiene un retardo de cero hacia *A*. *B* crea entonces una entrada en su tabla de enrutamiento para indicar que *A* está a un salto de distancia hacia la izquierda. Los demás enrutadores aún detectan que *A* está desactivado. En este punto, las entradas de la tabla de enrutamiento para *A* se muestran en la segunda fila de la figura 5-10(a). Durante el siguiente intercambio, *C* se entera de que *B* tiene una ruta a *A* de longitud 1, por lo que actualiza su tabla de enrutamiento para indicar una ruta de longitud 2, pero *D* y *E* no se enteran de las buenas nuevas sino hasta después. Como es evidente, las buenas noticias se difunden a razón de un salto por intercambio. En una red cuya ruta mayor tiene una longitud de *N* saltos, en un lapso de *N* intercambios todo mundo sabrá sobre los enlaces y enrutadores que acaban de revivir.

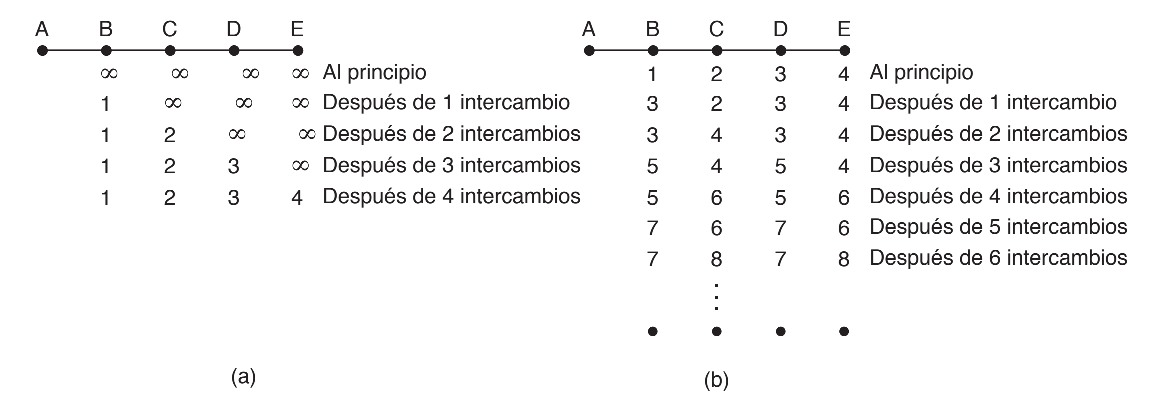


Figura 5-10. El problema del conteo al infinito.

Ahora consideremos la situación de la figura 5-10(b), en la que todos los enlaces y enrutadores están inicialmente activos. Los enrutadores *B*, *C*, *D* y *E* tienen distancias a *A* de 1, 2, 3 y 4 saltos, respectivamente. De pronto, *A* se desactiva o bien se corta el enlace entre *A* y *B* (que de hecho es lo mismo desde el punto de vista de *B*).

En el primer intercambio de paquetes, *B* no escucha nada de *A*. Por fortuna, *C* dice: "No te preocupes. Tengo una ruta hacia *A* de longitud 2". *B* no sabe que la ruta de *C* pasa a través de *B* mismo. Hasta donde *B* sabe, *C* puede tener 10 líneas, todas con rutas independientes hacia *A* de longitud 2. Como resultado, ahora *B* piensa que puede llegar a *A* por medio de *C*, con una longitud de ruta de 3. *D* y *E* no actualizan sus entradas para *A* en el primer intercambio.

En el segundo intercambio, *C* nota que cada uno de sus vecinos afirma tener una ruta de longitud 3 hacia *A*. *C* escoge una de ellas al azar y hace que su nueva distancia hacia *A* sea de 4, como se muestra en la tercera fila de la figura 5-10(b). Los intercambios subsecuentes producen la historia que se muestra en el resto de la figura 5-10(b).

A partir de esta figura debe quedar clara la razón por la que las malas noticias viajan con lentitud: no hay ningún enrutador que tenga en algún momento un valor mayor en más de una unidad que el mínimo de todos sus vecinos. Gradualmente, todos los enrutadores elevan su cuenta hacia el infinito, pero el número de intercambios requerido depende del valor numérico usado para el infinito. Por esta razón, es prudente hacer que el infinito sea igual a la ruta más larga, más 1.

No es del todo sorprendente que a éste se le conozca como el problema del **conteo al infinito**. Se han dado muchos intentos por resolverlo; por ejemplo, evitar que los enrutadores anuncien sus mejores rutas de vuelta a los vecinos de quienes las escucharon mediante la regla del horizonte dividido con envenenamiento en reversa que se describe en el RFC 1058. Sin embargo, ninguna de estas heurísticas funciona bien en la práctica a pesar de los nombres tan coloridos. El núcleo del problema es que, cuando *X* dice a *Y* que tiene una ruta hacia alguna parte, *Y* no tiene forma de saber si él mismo está en esa ruta.

5.2.5 Enrutamiento por estado del enlace

El enrutamiento por vector de distancia se utilizó en ARPANET hasta 1979, cuando se reemplazó por el enrutamiento por estado del enlace. El principal problema que provocó su desaparición era que, con frecuencia, el algoritmo tardaba demasiado en converger una vez que cambiaba la topología de la red (debido al problema del conteo al infinito). En consecuencia, se reemplazó por un algoritmo totalmente

nuevo, ahora conocido como **enrutamiento por estado del enlace**. Las variantes de este enrutamiento llamadas IS-IS y OSPF son los algoritmos de enrutamiento más utilizados dentro de las redes extensas y de Internet en la actualidad.

La idea detrás del enrutamiento por estado del enlace es bastante simple y se puede enunciar en cinco partes. Cada enrutador debe realizar lo siguiente para hacerlo funcionar:

1. Descubrir a sus vecinos y conocer sus direcciones de red.
2. Establecer la métrica de distancia o de costo para cada uno de sus vecinos.
3. Construir un paquete que indique todo lo que acaba de aprender.
4. Enviar este paquete a todos los demás enrutadores y recibir paquetes de ellos.
5. Calcular la ruta más corta a todos los demás enrutadores.

De hecho, se distribuye la topología completa a todos los enrutadores. Después se puede ejecutar el algoritmo de Dijkstra en cada enrutador para encontrar la ruta más corta a los demás enrutadores. A continuación veremos con mayor detalle cada uno de estos cinco pasos.

Aprender sobre los vecinos

Cuando un enrutador se pone en funcionamiento, su primera tarea es averiguar quiénes son sus vecinos. Para lograr esto envía un paquete especial HELLO en cada línea punto a punto. Se espera que el enrutador del otro extremo regrese una respuesta en la que indique su nombre. Estos nombres deben ser globalmente únicos puesto que, cuando un enrutador distante escucha después que hay tres enrutadores conectados a *F*, es indispensable que pueda determinar si los tres se refieren al mismo *F*.

Cuando se conectan dos o más enrutadores mediante un enlace de difusión (por ejemplo, un switch, anillo o Ethernet clásica), la situación es un poco más complicada. En la figura 5-11(a) se ilustra una LAN de difusión a la que están conectados de forma directa tres enrutadores, *A*, *C* y *F*. Cada uno de estos enrutadores está conectado a uno o más enrutadores adicionales, como se muestra.

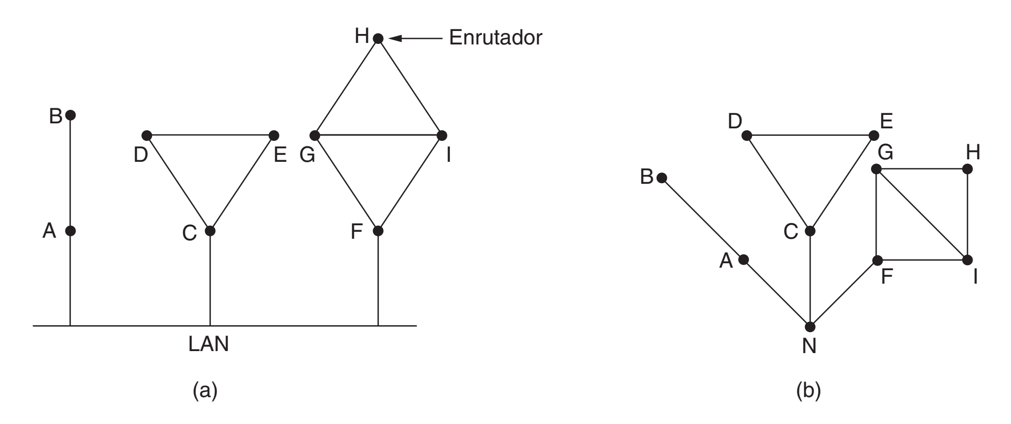


Figura 5-11. (a) Nueve enrutadores y una LAN de difusión. (b) Modelo de grafo de (a).

La LAN de difusión proporciona conectividad entre cada par de enrutadores conectados. Sin embargo, al modelar la LAN como muchos enlaces punto a punto se incrementa el tamaño de la topología, lo cual conduce a mensajes desperdiciados. Una mejor manera de modelar la LAN, es considerarla como un nodo más, como se muestra en la figura 5-11(b). Aquí hemos introducido un nodo artificial nuevo, *N*, al

que están conectados *A*, *C* y *F*. Se selecciona un **enrutador designado** en la LAN para que desempeñe el papel de *N* en el protocolo de enrutamiento. El hecho de que sea posible ir de *A* a *C* en la LAN se representa aquí mediante la ruta *ANC*.

Establecimiento de los costos de los enlaces

El algoritmo de enrutamiento por estado del enlace requiere que cada enlace tenga una métrica de distancia o costo para encontrar las rutas más cortas. El costo para llegar a los vecinos se puede establecer de modo automático, o el operador de red lo puede configurar. Una elección común es hacer el costo inversamente proporcional al ancho de banda del enlace. Por ejemplo, una red Ethernet de 1 Gbps puede tener un costo de 1 y una red Ethernet de 100 Mbps un costo de 10. Esto hace que las rutas de mayor capacidad sean mejores opciones.

Si la red está geográficamente dispersa, el retardo de los enlaces se puede considerar en el costo, de modo que las rutas a través de enlaces más cortos sean mejores opciones. La manera más directa de determinar este retardo es enviar un paquete especial ECO a través de la línea, que el otro extremo tendrá que regresar de inmediato. Si se mide el tiempo de ida y vuelta, y se divide entre dos, el enrutador emisor puede obtener una estimación razonable del retardo.

Construcción de los paquetes de estado del enlace

Una vez que se ha recabado la información necesaria para el intercambio, el siguiente paso es que cada enrutador construya un paquete que contenga todos los datos. El paquete comienza con la identidad del emisor, seguida de un número de secuencia, una edad (que describiremos después) y una lista de vecinos. También se proporciona el costo para cada vecino. En la figura 5-12(a) se muestra una red de ejemplo; los costos se muestran como etiquetas en las líneas. En la figura 5-12(b) se muestran los paquetes de estado del enlace correspondientes para los seis enrutadores.

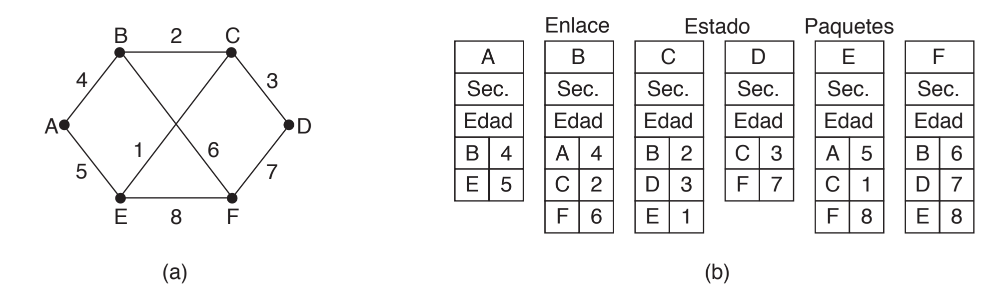


Figura 5-12. (a) Una red. (b) Los paquetes de estado del enlace para esta red.

Es fácil construir los paquetes de estado del enlace. La parte difícil es determinar cuándo construirlos. Una posibilidad es construirlos de manera periódica; es decir, a intervalos regulares. Otra posibilidad es construirlos cuando ocurra un evento significativo, como la caída o la reactivación de una línea o de un vecino, o cuando sus propiedades cambien en forma considerable.

Distribución de los paquetes de estado del enlace

La parte más complicada del algoritmo es la distribución de los paquetes de estado del enlace. Todos los enrutadores deben recibir todos los paquetes de estado del enlace con rapidez y confiabilidad. Si se utili-

zan distintas versiones de la topología, las rutas que se calculen podrían tener inconsistencias como ciclos, máquinas inalcanzables y otros problemas.

Primero describiremos el algoritmo básico de distribución. Después le aplicaremos algunos refinamientos. La idea fundamental es utilizar inundación para distribuir los paquetes de estado del enlace a todos los enrutadores. Con el fin de mantener controlada la inundación, cada paquete contiene un número de secuencia que se incrementa con cada nuevo paquete enviado. Los enrutadores llevan el registro de todos los pares (enrutador de origen, secuencia) que ven. Cuando llega un nuevo paquete de estado del enlace, se verifica y compara con la lista de paquetes ya vistos. Si es nuevo, se reenvía a través de todas las líneas, excepto aquella por la que llegó. Si es un duplicado, se descarta. Si llega un paquete con número de secuencia menor que el mayor visto hasta el momento, se rechaza como obsoleto debido a que el enrutador tiene datos más recientes.

Este algoritmo tiene algunos problemas, pero son manejables. Primero, si los números de secuencia vuelven a comenzar, reinará la confusión. La solución aquí es utilizar un número de secuencia de 32 bits. Con un paquete de estado del enlace por segundo, el tiempo para volver a empezar será de 137 años, por lo que podemos ignorar esta posibilidad.

Segundo, si llega a fallar un enrutador, perderá el registro de su número de secuencia. Si comienza nuevamente en 0, se rechazará como duplicado el siguiente paquete que envíe.

Tercero, si llega a corromperse un número de secuencia y se recibe 65 540 en vez de 4 (un error de 1 bit), los paquetes 5 a 65 540 se rechazarán como obsoletos, dado que se piensa que el número de secuencia actual es 65 540.

La solución a todos estos problemas es incluir la edad de cada paquete después del número de secuencia y disminuirla una vez cada segundo. Cuando la edad llega a cero, se descarta la información de ese enrutador. Por lo general, un paquete nuevo entra, por ejemplo, cada 10 segundos, por lo que la información de los enrutadores sólo expira cuando un enrutador está caído (o cuando se pierden seis paquetes consecutivos, un evento poco probable). Los enrutadores también decrementan el campo *Edad* durante el proceso inicial de inundación para asegurar que no pueda perderse ningún paquete y sobrevivir durante un periodo de tiempo indefinido (se descarta el paquete cuya edad sea cero).

Algunos refinamientos a este algoritmo pueden hacerlo más robusto. Una vez que llega un paquete de estado del enlace a un enrutador para ser inundado, no se encola para su transmisión de inmediato, sino que se coloca en un área de almacenamiento para esperar un tiempo corto, en caso de que se activen o desactiven más enlaces. Si llega otro paquete de estado del enlace proveniente del mismo origen antes de que se transmita el primer paquete, se comparan sus números de secuencia. Si son iguales, se descarta el duplicado. Si son diferentes, se desecha el más antiguo. Como protección contra los errores en los enlaces, se confirma la recepción de todos los paquetes de estado del enlace.

En la figura 5-13 se describe la estructura de datos que utiliza el enrutador *B* para la red de la figura 5-12(a). Cada fila aquí corresponde a un paquete de estado del enlace recién llegado, pero que aún no se procesa por completo. La tabla registra dónde se originó el paquete, su número de secuencia y edad, así como los datos. Además, hay banderas de envío y de confirmación de recepción para cada uno de los tres enlaces de *B* (a *A*, *C* y *F*, respectivamente). Las banderas de envío significan que el paquete debe enviarse a través del enlace indicado. Las banderas de confirmación de recepción significan que ahí se debe confirmar su recepción.

En la figura 5-13, el paquete de estado del enlace de *A* llega de manera directa, por lo que se debe enviar a *C* y *F*, y debe confirmarse la recepción a *A*, como lo indican los bits de bandera. De la misma forma, es necesario reenviar el paquete de *F* a *A* y a *C*, y debe enviarse a *F* la confirmación de su recepción.

Sin embargo, la situación del tercer paquete, que proviene de *E*, es diferente. Llega dos veces, la primera a través de *EAB* y la segunda por medio de *EFB*. En consecuencia, este paquete tiene que enviarse sólo a *C*, pero se debe confirmar su recepción tanto a *A* como a *F*, como lo indican los bits.

Origen	Sec.	Edad	Banderas de envío A	C	F	Banderas de ACK A	C	F	Datos
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

Figura 5-13. El búfer de paquetes para el enrutador *B* de la figura 5-12(a).

Si llega un duplicado mientras el original aún está en el búfer, se deben cambiar los bits. Por ejemplo, si llega una copia del estado de *C* desde *F* antes de que la cuarta entrada en la tabla haya sido reenviada, cambiarán los 6 bits a 100011 para indicar que se debe enviar a F una confirmación de recepción del paquete, pero sin enviarle el paquete mismo.

Cálculo de las nuevas rutas

Una vez que un enrutador ha acumulado un conjunto completo de paquetes de estado del enlace, puede construir el grafo de toda la red debido a que todos los enlaces están simbolizados. De hecho, cada enlace se representa dos veces, una para cada dirección. Las distintas direcciones pueden tener incluso costos diferentes. Así, los cálculos de la ruta más corta pueden encontrar rutas del enrutador *A* a *B* que sean distintas a las del enrutador de *B* a *A*.

Ahora se puede ejecutar localmente el algoritmo de Dijkstra para construir las rutas más cortas a todos los destinos posibles. Los resultados de este algoritmo indican al enrutador qué enlace debe usar para llegar a cada destino. Esta información se instala en las tablas de enrutamiento y se puede reanudar la operación normal.

En comparación con el enrutamiento por vector de distancia, el enrutamiento por estado del enlace requiere más memoria y poder de cómputo. Para una red con *n* enrutadores, cada uno de los cuales tiene *k* vecinos, la memoria requerida para almacenar los datos de entrada es proporcional a *kn*, que es por lo menos tan grande como una tabla de enrutamiento que lista todos los destinos. Además, el tiempo de cómputo también crece con más rapidez que *kn*, incluso con las estructuras de datos más eficientes; un problema en las grandes redes. Sin embargo, en muchas situaciones prácticas, el enrutamiento por estado del enlace funciona bien debido a que no sufre de los problemas de convergencia lenta.

El enrutamiento por estado del enlace se usa ampliamente en las redes actuales, por lo que son pertinentes unas pocas palabras sobre algunos protocolos que lo usan. Muchos ISP usan el protocolo de estado del enlace **IS-IS** (**Sistema Intermedio a Sistema Intermedio**, del inglés *Intermediate System-Intermediate System*) (Oran, 1990). Este protocolo fue diseñado para una de las primeras redes llamada DECnet; más tarde lo adoptó la ISO para utilizarlo con los protocolos OSI y después se modificó para manejar otros protocolos también, siendo IP el más importante de éstos. El otro protocolo de estado es **OSPF** (**Abrir primero la ruta más corta**, del inglés *Open Shortest Path First*), diseñado por el IETF varios años después de IS-IS y adoptó muchas de las innovaciones diseñadas para este último. Estas innovaciones incluyen un método de estabilización automática para inundar las actualizaciones de estado del enlace, el concepto de un enrutador designado en una LAN y el método para calcular y soportar la división de rutas y múltiples métricas. Como consecuencia, hay muy poca diferencia entre IS-IS y OSPF.

La diferencia más importante es que IS-IS puede transportar información sobre varios protocolos de capa de red al mismo tiempo (por ejemplo, IP, IPX y AppleTalk). OSPF no tiene esta característica y es una ventaja en los grandes entornos multiprotocolo. En la sección 5.6.6 estudiaremos el OSPF.

También es pertinente hacer un comentario general sobre los algoritmos de enrutamiento. Los algoritmos de estado del enlace, de vector de distancia y otros más dependen del procesamiento en todos los enrutadores para calcular las rutas. Los problemas con el hardware o el software, incluso en una pequeña cantidad de enrutadores, pueden causar estragos en la red. Por ejemplo, si un enrutador afirma tener un enlace que no tiene u olvida un enlace que sí tiene, el grafo de la red será incorrecto. Si un enrutador deja de reenviar paquetes, o los corrompe al hacerlo, la ruta no funcionará como se esperaba. Por último, si al enrutador se le acaba la memoria o se ejecuta mal el cálculo de enrutamiento, surgirán problemas. A medida que la red crece en decenas o cientos de miles de nodos, la probabilidad de que un enrutador tenga fallas ocasionales deja de ser insignificante. El truco es tratar de limitar el daño cuando ocurra lo inevitable. Perlman (1988) analiza estos problemas y sus soluciones con detalle.

5.2.6 Enrutamiento jerárquico

A medida que crece el tamaño de las redes, también lo hacen en forma proporcional las tablas de enrutamiento del enrutador. Las tablas que están en crecimiento constante no sólo consumen memoria del enrutador, sino que también se necesita más tiempo de CPU para examinarlas y más ancho de banda para enviar informes de estado entre enrutadores. En cierto momento, la red puede crecer hasta el punto en que ya no sea viable que cada enrutador tenga una entrada para cada uno de los demás enrutadores, por lo que el enrutamiento tendrá que hacerse de manera jerárquica, como ocurre en la red telefónica.

Cuando se utiliza el enrutamiento jerárquico, los enrutadores se dividen en lo que llamaremos **regiones**. Cada enrutador conoce todos los detalles para enrutar paquetes a destinos dentro de su propia región, pero no sabe nada de la estructura interna de las otras regiones. Cuando se interconectan diferentes redes, es natural considerar cada una como región independiente con el fin de liberar a los enrutadores de una red de la necesidad de conocer la estructura topológica de las demás redes.

En las redes enormes, tal vez no sea suficiente una jerarquía de dos niveles; puede ser necesario agrupar las regiones en clústeres, los clústeres en zonas, las zonas en grupos, etc, hasta que se nos agoten los nombres para clasificarlos. Como ejemplo de jerarquía multinivel, considere una posible forma de enrutar un paquete de Berkeley, California, a Malindi, Kenia. El enrutador de Berkeley podría conocer la topología detallada de California, pero tendría que enviar todo el tráfico exterior al enrutador de Los Ángeles. El enrutador de Los Ángeles podría enrutar el tráfico directamente a otros enrutadores del país, pero tendría que enviar el tráfico internacional a Nueva York. El enrutador de Nueva York podría estar programado para dirigir todo el tráfico al enrutador del país de destino encargado del manejo de tráfico internacional, por decir, en Nairobi. Por último, el paquete encontraría su camino por el árbol de Kenia hasta llegar a Malindi.

En la figura 5-14 aparece un ejemplo cuantitativo de enrutamiento en una jerarquía de dos niveles con cinco regiones. La tabla de enrutamiento completa para el enrutador *1A* tiene 17 entradas, como se muestra en la figura 5-14(b). Si el enrutamiento se hace en forma jerárquica, como en la figura 5-14(c), hay entradas para todos los enrutadores locales, igual que antes, pero las demás regiones se condensan en un solo enrutador, de modo que todo el tráfico para la región 2 viaja a través de la línea *1B-2A*, pero el resto del tráfico remoto viaja por la línea *1C-3B*. El enrutamiento jerárquico redujo la tabla de 17 entradas a 7. A medida que crece la razón entre la cantidad de regiones y el número de enrutadores por región, aumentan los ahorros de espacio en la tabla.

Por desgracia, estas ganancias de espacio no son gratuitas. Hay que pagar un precio: un incremento en la longitud de la ruta. Por ejemplo, la mejor ruta de *1A* a *5C* es a través de la región 2, pero con el

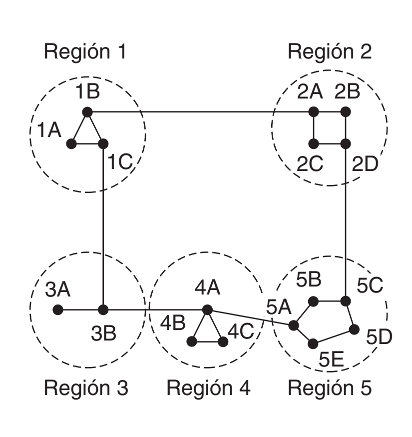


Tabla completa para 1A

Dest.	Línea	Saltos
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

(b)

Tabla jerárquica para 1A

Dest.	Línea	Saltos
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

(c)

Figura 5-14. Enrutamiento jerárquico.

enrutamiento jerárquico, todo el tráfico a la región 5 pasa a través de la región 3, porque eso es mejor para la mayoría de los destinos de la región 5.

Cuando una sola red se vuelve muy grande, surge una pregunta interesante: ¿cuántos niveles debe tener la jerarquía? Por ejemplo, considere una red con 720 enrutadores. Si no hay jerarquía, cada enrutador necesita 720 entradas en la tabla de enrutamiento. Si dividimos la red en 24 regiones de 30 enrutadores cada una, cada enrutador necesitará 30 entradas locales más 23 entradas remotas, lo que da un total de 53 entradas. Si elegimos una jerarquía de tres niveles con 8 clústeres, cada uno de los cuales contiene 9 regiones de 10 enrutadores, cada enrutador necesita 10 entradas para los enrutadores locales; 8, para el enrutamiento a otras regiones dentro de su propio clúster, y 7, para clústeres distantes, lo que da un total de 25 entradas. Kamoun y Kleinrock (1979) descubrieron que el número óptimo de niveles para una red de N enrutadores es de ln N, y se requieren un total de e ln N entradas por enrutador. Ellos también demostraron que el aumento en la longitud promedio efectiva de la ruta causado por el enrutamiento jerárquico es tan pequeño que por lo general es aceptable.

5.2.7 Enrutamiento por difusión

En algunas aplicaciones, los hosts necesitan enviar mensajes a varios o a todos los hosts en la red. Por ejemplo, el servicio de distribución de informes sobre el clima, la actualización de los precios de la bolsa o los programas de radio en vivo podrían funcionar mejor si se difunden a todas las máquinas para dejar que las personas interesadas lean los datos. El envío simultáneo de un paquete a todos los destinos se llama **difusión** (*broadcasting*). Se han propuesto varios métodos para llevarla a cabo.

Un método de difusión que no requiere características especiales de la red es que el origen sólo envíe un paquete distinto a cada destino. El método no sólo desperdicia ancho de banda y es lento, sino que

también requiere que el origen tenga una lista completa de todos los destinos. En la práctica este método no es deseable, aun cuando tiene muchas aplicaciones.

Una mejora del algoritmo anterior es el **enrutamiento multidestino**, en donde cada paquete contiene una lista de destinos o un mapa de bits que indica los destinos deseados. Cuando un paquete llega al enrutador, éste revisa todos los destinos para determinar el conjunto de líneas de salida que necesitará (se requiere una línea de salida si es la mejor ruta a cuando menos uno de los destinos). El enrutador genera una nueva copia del paquete para cada línea de salida que se utilizará, e incluye en cada paquete sólo aquellos destinos que utilizarán la línea. En efecto, el grupo de destinos se divide entre las líneas de salida. Después de una cantidad suficiente de saltos, cada paquete llevará sólo un destino, como un paquete normal. El enrutamiento multidestino es como los paquetes con direccionamiento individual, sólo que cuando varios paquetes deben seguir la misma ruta, uno de ellos paga la tarifa completa y los demás viajan gratis. Por lo tanto, el ancho de banda se utiliza con más eficiencia. Sin embargo, este esquema aún requiere que el origen conozca todos los destinos, además de que representa el mismo trabajo para un enrutador determinar a dónde debe enviar un paquete multidestino que varios paquetes distintos.

Ya hemos visto una mejor técnica de enrutamiento por difusión: la inundación. Cuando se implementa con un número de secuencia por cada origen, la inundación usa los enlaces de manera eficiente con una regla de decisión en los enrutadores que es relativamente simple. Aunque la inundación es poco adecuada para la comunicación punto a punto ordinaria, para difusión puede merecer que se le considere con seriedad. Sin embargo, resulta ser que podemos hacer algo todavía mejor una vez que se han calculado las rutas más cortas para los paquetes regulares.

La idea del **reenvío por ruta invertida** (*reverse path forwarding*) es elegante y excepcionalmente sencilla una vez planteada (Dalal y Metcalfe, 1978). Cuando llega un paquete difundido a un enrutador, éste lo revisa para ver si llegó por el enlace que se usa por lo común para enviar paquetes *hacia* el origen de la difusión. De ser así, hay excelentes posibilidades de que el paquete difundido haya seguido la mejor ruta desde el enrutador y, por lo tanto, sea la primera copia en llegar al enrutador. Si éste es el caso, el enrutador reenvía copias del paquete a todos los enlaces, excepto a aquel por el que llegó. No obstante, si el paquete difundido llegó por un enlace diferente del preferido para llegar al origen, el paquete se descarta como probable duplicado.

En la figura 5-15 se muestra un ejemplo del reenvío por ruta invertida. En la parte (a) se muestra una red; en la parte (b), un árbol sumidero para el enrutador *I* de esa red y en la parte (c), el funcionamiento del algoritmo de ruta invertida. En el primer salto, *I* envía paquetes a *F*, *H*, *J* y *N*, como lo indica la segunda fila del árbol. Cada uno de estos paquetes llega a *I* por la ruta preferida (suponiendo que la ruta preferida pasa a través del árbol sumidero), como lo indica el círculo alrededor de la letra. En el segundo salto

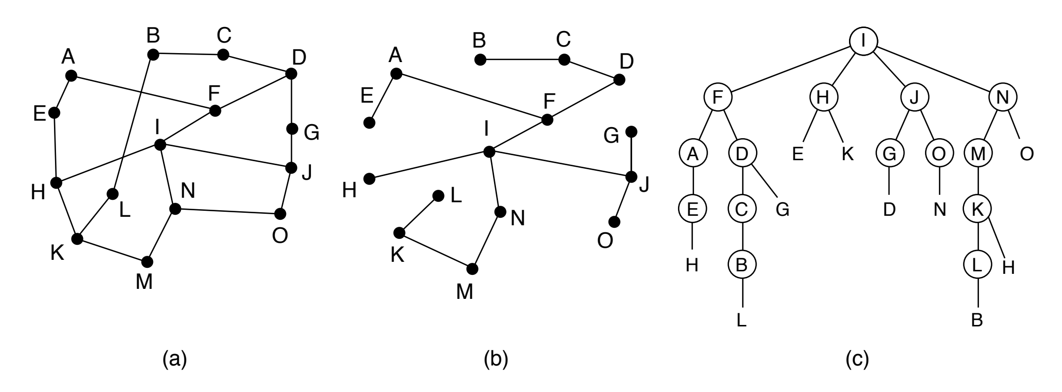


Figura 5-15. Reenvío por ruta invertida. (a) Una red. (b) Un árbol sumidero. (c) El árbol construido mediante reenvío por ruta invertida.

se generan ocho paquetes, dos por cada uno de los enrutadores que recibieron un paquete en el primer salto. Como resultado los ocho llegan a enrutadores no visitados con anterioridad, y cinco llegan a través de la línea preferida. De los seis paquetes generados en el tercer salto, sólo tres llegan por la ruta preferida (a *C*, *E* y *K*); los otros son duplicados. Después de cinco saltos y 24 paquetes, termina la difusión, en comparación con cuatro saltos y 14 paquetes si se hubiera seguido exactamente el árbol sumidero.

La ventaja principal del reenvío por ruta invertida es que es tanto eficiente como fácil de implementar. Envía el paquete de difusión a través de cada enlace sólo una vez en cada dirección, como en la inundación, pero sólo requiere que los enrutadores sepan cómo llegar a todos los destinos, sin que necesiten recordar los números de secuencia (o usar otros mecanismos para detener la inundación) o listar todos los destinos en el paquete.

Nuestro último algoritmo de difusión mejora el comportamiento del reenvío por ruta invertida. Usa de manera explícita el árbol sumidero (o cualquier otro árbol de expansión conveniente) para el enrutador que inicia la difusión. Un **árbol de expansión** es un subconjunto de la red que incluye todos los enrutadores pero no contiene ciclos. Los árboles sumidero son árboles de expansión. Si cada enrutador sabe cuáles de sus líneas pertenecen al árbol de expansión, puede copiar un paquete de difusión entrante en todas las líneas del árbol de expansión, excepto en aquella por la que llegó. Este método utiliza de manera óptima el ancho de banda, ya que genera el número mínimo absoluto de paquetes necesarios para llevar a cabo el trabajo. En la figura 5-15, por ejemplo, cuando el árbol de sumidero de la parte (b) es usado como el árbol de expansión, los paquetes de difusión son enviados con un mínimo de 14 paquetes. El único problema es que cada enrutador debe tener conocimiento de algún árbol de expansión para que este método pueda funcionar. Algunas veces esta información está disponible (por ejemplo, con el enrutamiento por estado del enlace, todos los enrutadores conocen la topología completa, por lo que pueden calcular un árbol de expansión), pero a veces no (por ejemplo, con el enrutamiento por vector de distancia).

5.2.8 Enrutamiento multidifusión

Algunas aplicaciones, como un juego multijugador o un video en vivo de un evento deportivo que se transmite por flujo continuo a muchas ubicaciones, envían paquetes a múltiples receptores. A menos que el grupo sea muy pequeño, es costoso enviar un paquete distinto a cada receptor. Por otro lado, sería un desperdicio difundir un paquete si el grupo consiste en, por decir, 1000 máquinas en una red con un millón de nodos, de tal forma que la mayoría de los receptores no están interesados en el mensaje (o peor aún, están en definitiva interesados pero se supone que no deben verlo). Por lo tanto, necesitamos una manera de enviar mensajes a grupos bien definidos que sean grandes en número, pero pequeños en comparación con la totalidad de la red.

El proceso de enviar un mensaje a uno de tales grupos se denomina **multidifusión** (*multicasting*); el algoritmo de enrutamiento que se utiliza es el **enrutamiento por multidifusión**. Todos los esquemas de multidifusión requieren alguna forma de crear y destruir grupos, además de identificar qué enrutadores son miembros de un grupo. La forma de realizar estas tareas no le concierne al algoritmo de enrutamiento. Por ahora vamos a suponer que cada grupo se identifica mediante una dirección de multidifusión y que los enrutadores conocen los grupos a los que pertenecen. Volveremos a tocar el tema de la membresía de grupo cuando estudiemos la capa de red de Internet en la sección 5.6.

Los esquemas de enrutamiento por multidifusión se basan en los esquemas de enrutamiento por difusión que ya estudiamos; envían paquetes a través de árboles de expansión para entregar esos paquetes a los miembros del grupo, al tiempo que optimizan el uso del ancho de banda. Sin embargo, el mejor árbol de expansión a usar depende de si el grupo es denso, con receptores esparcidos por toda la red o disperso, en donde gran parte de la red no pertenece al grupo. En esta sección consideraremos ambos casos.

Si el grupo es denso, la difusión es un buen comienzo debido a que transmite de manera eficiente el paquete a todas las partes de la red. Pero la difusión llegará a algunos enrutadores que no sean miembros del grupo, lo cual es un desperdicio. La solución explorada por Deering y Cheriton (1990) es recortar el árbol de expansión de difusión, mediante la eliminación de enlaces que no conducen a los miembros. El resultado es un árbol de expansión de multidifusión eficiente.

Como ejemplo, considere los dos grupos, 1 y 2, en la red que se muestra en la figura 5-16(a). Algunos enrutadores están conectados a hosts que pertenecen a uno o ambos grupos, como se indica en la figura. En la figura 5-16(b) se muestra un árbol de expansión para el enrutador de la izquierda. Este árbol se puede usar para difusión, pero es exagerado para la multidifusión, como podemos ver de las dos versiones recortadas que se muestran a continuación. En la figura 5-16(c) se han eliminado todos los enlaces que no conducen a hosts que sean miembros del grupo 1. El resultado es el árbol de expansión de multidifusión para el enrutador de la izquierda. Los paquetes se reenvían sólo a través de este árbol de expansión, lo cual es más eficiente que el árbol de difusión debido a que hay 7 enlaces en vez de 10. En la figura 5-16(d) se muestra el árbol de expansión de multidifusión después de hacer recortes para el grupo 2. También es eficiente, con sólo cinco enlaces esta vez. De igual forma muestra que los diferentes grupos de multidifusión tienen distintos árboles de expansión.

Hay varias maneras de recortar el árbol de expansión. Se puede utilizar la más sencilla si se maneja el enrutamiento por estado del enlace, y cada enrutador está consciente de la topología completa, incluyendo qué hosts pertenecen a cuáles grupos. Así, cada enrutador puede construir su propio árbol de expansión recortado para cada emisor que envía datos al grupo en cuestión, para lo cual construye un árbol de expansión para el emisor de la manera usual y después elimina todos los enlaces que no conectan a los miembros del grupo con el nodo sumidero. **MOSPF** (**OSPF de Multidifusión**, del inglés *Multicast OSPF*) es un ejemplo de un protocolo de estado del enlace que funciona de esta manera (Moy, 1994).

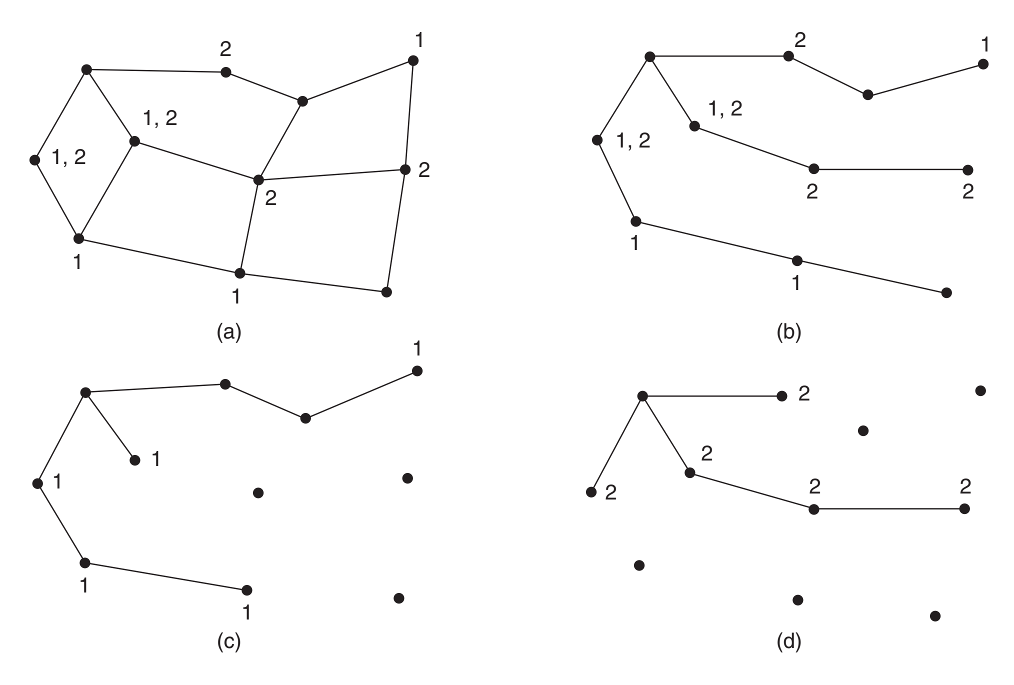


Figura 5-16. (a) Una red. (b) Un árbol de expansión para el enrutador del extremo izquierdo. (c) Un árbol de multidifusión para el grupo 1. (d) Un árbol de multidifusión para el grupo 2.

Con el enrutamiento por vector de distancia se puede seguir una estrategia de recorte diferente. El algoritmo básico es el reenvío por ruta invertida. Sin embargo, cuando un enrutador sin hosts interesados en un grupo en particular y sin conexiones con otros enrutadores recibe un mensaje de multidifusión para ese grupo, responde con un mensaje PRUNE (de recorte) para indicar al vecino emisor que no envíe más multidifusiones para ese grupo. Cuando un enrutador que no tiene miembros del grupo entre sus propios hosts recibe uno de tales mensajes por todas las líneas a las que envía la multidifusión, también puede responder con un mensaje PRUNE. De esta forma, el árbol de expansión se recorta recursivamente. **DVMRP** (**Protocolo de Enrutamiento Multidifusión de Vector de Distancia**, del inglés *Distance Vector Multicast Routing Protocol*) es un ejemplo de un protocolo de enrutamiento multidifusión que funciona de esta manera (Waitzman y colaboradores, 1988).

El recorte produce árboles de expansión eficientes que sólo utilizan los enlaces realmente necesarios para llegar a los miembros del grupo. Una desventaja potencial de este algoritmo es que representa mucho trabajo para los enrutadores, en especial en redes grandes. Suponga que una red tiene *n* grupos, cada uno con un promedio de *m* nodos. En cada enrutador y por cada grupo se deben almacenar *m* árboles de expansión recortados, lo que da un total de *mn* árboles. Por ejemplo, la figura 5-16(c) muestra el árbol de expansión para que el enrutador del extremo izquierdo envíe datos al grupo 1. El árbol de expansión para que el enrutador del extremo derecho envíe datos al grupo 1 (que no se muestra) tendrá una apariencia bastante distinta, ya que los paquetes se dirigirán de manera directa a los miembros del grupo en vez de hacerlo a través del lado izquierdo del grafo. Esto a su vez significa que los enrutadores deben reenviar los paquetes destinados al grupo 1 en distintas direcciones, dependiendo de cuál nodo envíe datos al grupo. Cuando existen muchos grupos grandes con varios emisores, se requiere un almacenamiento considerable para almacenar todos los árboles.

Un diseño alterno utiliza **árboles basados en núcleo** (*core-based trees*) para calcular un solo árbol de expansión por grupo (Ballardie y colaboradores, 1993). Todos los enrutadores coinciden en una raíz (el **núcleo** o **punto de reunión**) y para construir el árbol, envían un paquete de cada miembro a la raíz. El árbol es la unión de las rutas trazadas por estos paquetes. La figura 5-17(a) muestra un árbol basado en núcleo para el grupo 1. Para enviar datos a este grupo, un emisor envía un paquete al núcleo. Cuando el paquete llega al núcleo, se reenvía hacia abajo por el árbol. Esto se muestra en la figura 5-17(b) para el emisor del lado derecho de la red. Para optimizar el desempeño, los paquetes destinados para el grupo no necesitan llegar al núcleo antes de que se envíen por multidifusión. Tan pronto como un paquete llega al árbol, se puede reenviar hacia la raíz y por todas las demás ramas. Éste es el caso para el emisor en la parte superior de la figura 5-17(b).

Tener un árbol compartido no es óptimo para todas las fuentes. Por ejemplo, en la figura 5-17(b) el paquete del emisor en el extremo derecho llega al miembro del grupo de la parte superior derecha a tra-

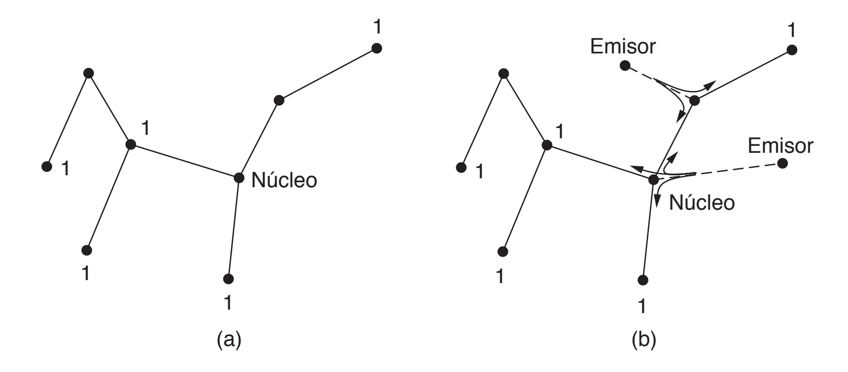


Figura 5-17. (a) Árbol basado en núcleo para el grupo 1. (b) Envío de datos al grupo 1.

vés del núcleo en tres saltos, en vez de hacerlo de manera directa. La ineficiencia depende de la ubicación del núcleo y los emisores, pero a menudo es razonable cuando el núcleo está en medio de ellos. Cuando sólo hay un emisor, como en un video que se transmite a un grupo por flujo continuo, es óptimo usar el emisor como el núcleo.

Cabe mencionar también que los árboles compartidos pueden representar un ahorro importante en los costos de almacenamiento, los mensajes enviados y el poder de cómputo. Cada enrutador sólo tiene que mantener un árbol por grupo, en vez de *m* árboles. Además, los enrutadores que no forman parte del árbol no hacen ningún esfuerzo por apoyar el grupo. Por esta razón, los métodos de árbol compartido como los árboles basados en núcleo se utilizan para la multidifusión hacia grupos dispersos en Internet, como parte de protocolos populares como **PIM** (**Multidifusión Independiente del Protocolo**, del inglés *Protocol Independent Multicast*) (Fenner y colaboradores, 2006).

5.2.9 Enrutamiento anycast

Hasta ahora hemos cubierto los modelos de distribución en los que un origen envía a un solo destino (**unidifusión** o *unicast*), a todos los destinos (difusión o *broadcast*) y a un grupo de destinos (multidifusión o *multicast*). Hay otro modelo de distribución llamado **anycast**, que algunas veces también es útil. En anycast, un paquete se entrega al miembro más cercano de un grupo (Partridge y colaboradores, 1993). Los esquemas que encuentran estas rutas se denominan **enrutamiento anycast**.

¿Por qué querríamos usar anycast? En ocasiones los nodos proporcionan un servicio, como la hora del día o la distribución de contenido, en donde todo lo que importa es obtener la información correcta sin importar el nodo con el que se haya hecho contacto; cualquiera es igual. Por ejemplo, anycast se utiliza en Internet como parte del servicio DNS, como veremos en el capítulo 7.

Por suerte no tenemos que idear nuevos esquemas de enrutamiento para anycast, ya que los enrutamientos por vector de distancia y de estado del enlace regulares pueden producir rutas anycast. Suponga que deseamos enviar datos mediante anycast a los miembros del grupo 1. Todos recibirán la dirección "1" en lugar de distintas direcciones. El enrutamiento por vector de distancia distribuirá los vectores en la forma usual y los nodos elegirán la ruta más corta hacia el destino 1. Como resultado, los nodos enviarán datos a la instancia más cercana del destino 1. Las rutas se muestran en la figura 5-18(a). Este procedimiento funciona debido a que el protocolo de enrutamiento no se da cuenta de que hay varias instancias del destino 1. Es decir, cree que todas las instancias del nodo 1 son el mismo nodo, como en la topología que se muestra en la figura 5-18(b).

Este procedimiento funciona también para el enrutamiento de estado del enlace, aunque con la consideración adicional de que el protocolo de enrutamiento no debe encontrar rutas que parezcan ser las más

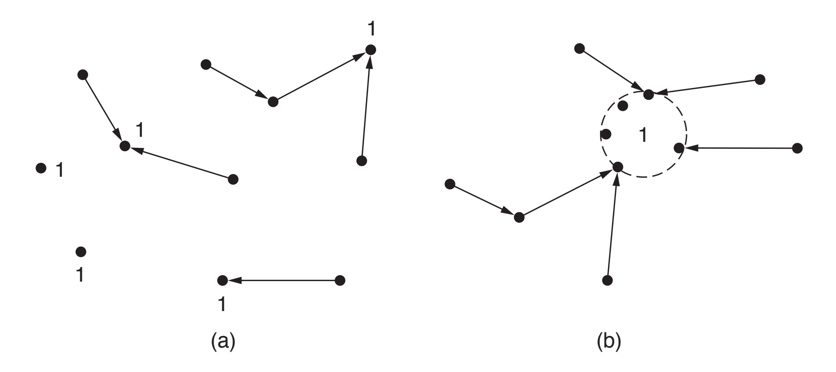


Figura 5-18. (a) Rutas anycast al grupo 1. (b) Topología que ve el protocolo de enrutamiento.

cortas y que pasen a través del nodo 1. Esto produciría saltos hacia el hiperespacio, ya que las instancias del nodo 1 en realidad son nodos ubicados en distintas partes de la red. Sin embargo, los protocolos de estado del enlace ya pueden hacer esta distinción entre los enrutadores y los hosts. Pasamos por alto este hecho antes debido a que no era necesario para nuestro estudio.

5.2.10 Enrutamiento para hosts móviles

Millones de personas usan computadoras mientras se trasladan de un lado a otro, desde situaciones verdaderamente móviles con dispositivos inalámbricos en autos en movimiento, hasta situaciones nómadas en las que se utilizan computadoras portátiles en una serie de distintas localidades. Usaremos el término **hosts móviles** para referirnos a cualquiera de las dos categorías antes mencionadas, para diferenciarlas de los hosts fijos que nunca se mueven. Cada vez es más común que las personas quieran estar conectadas en cualquier parte del mundo en donde se encuentren, con la misma facilidad que si estuvieran en su hogar. Estos hosts móviles introducen una nueva complicación: para enrutar un paquete a un host móvil, la red tiene qué encontrarlo primero.

El modelo del mundo que consideraremos es uno en el que se supone que todos los hosts tienen una **ubicación base** (*home location*) permanente que nunca cambia. Además, cada host tiene una dirección base (*home address*) permanente que se puede usar para determinar su ubicación base, algo parecido a la forma en que el número telefónico 1-212-5551212 indica que se encuentra en Estados Unidos (código de país 1) y Manhattan (212). La meta de enrutamiento en los sistemas con hosts móviles es posibilitar el envío de paquetes a hosts móviles mediante el uso de su dirección base, y hacer que los paquetes lleguen de manera eficiente a ellos en cualquier lugar en el que puedan estar. Lo difícil, por supuesto, es encontrarlos.

Aquí es pertinente mencionar unos detalles sobre este modelo. Un modelo diferente tendría que recalcular rutas mientras el host móvil se desplaza y cambia la topología. Entonces, simplemente usaríamos los esquemas de enrutamiento que describimos antes en esta sección. Sin embargo, con un número creciente de hosts móviles, la red, con este modelo pronto terminaría calculando nuevas rutas sin parar. Al usar las direcciones locales se reduce de manera considerable esta carga.

Otra alternativa sería proporcionar movilidad por encima de la capa de red, algo que ocurre comúnmente con las computadoras portátiles en la actualidad. Cuando se desplazan a nuevas ubicaciones de Internet, las computadoras portátiles adquieren nuevas direcciones de red. No hay asociación entre la dirección antigua y la nueva; la red no sabe que pertenecían a la misma computadora portáltil. En este modelo, podemos usar una computadora portátil para navegar en web, pero otros hosts no le pueden enviar paquetes (por ejemplo, para una llamada entrante) sin construir un servicio de ubicación en la capa superior; por ejemplo, *de nuevo* iniciar sesión en Skype después de moverse. Además, las conexiones no se pueden mantener mientras el host está en movimiento; hay que iniciar nuevas conexiones. La movilidad en la capa de red es útil para corregir estos problemas.

La idea básica que se usa para el enrutamiento móvil en las redes de Internet y celulares es que el host móvil indique su posición actual a un host que se encuentre en la ubicación base. Este host, que actúa en beneficio del host móvil, se denomina **agente de base** (*home agent*). Una vez que conoce la ubicación actual del host móvil, puede reenviar paquetes para que sean entregados.

La figura 5-19 muestra el enrutamiento móvil en acción. Un emisor al noroeste de la ciudad de Seattle desea enviar un paquete a un host que por lo general se encuentra al otro lado de Estados Unidos, en Nueva York. El caso de interés para nosotros es cuando el host móvil no se encuentra en su ubicación base; por ejemplo, que se encuentre temporalmente en San Diego.

El host móvil en San Diego debe adquirir una dirección de la red local antes de usar esta red. Esto ocurre de la manera usual en la que los hosts obtienen direcciones de red; más adelante veremos

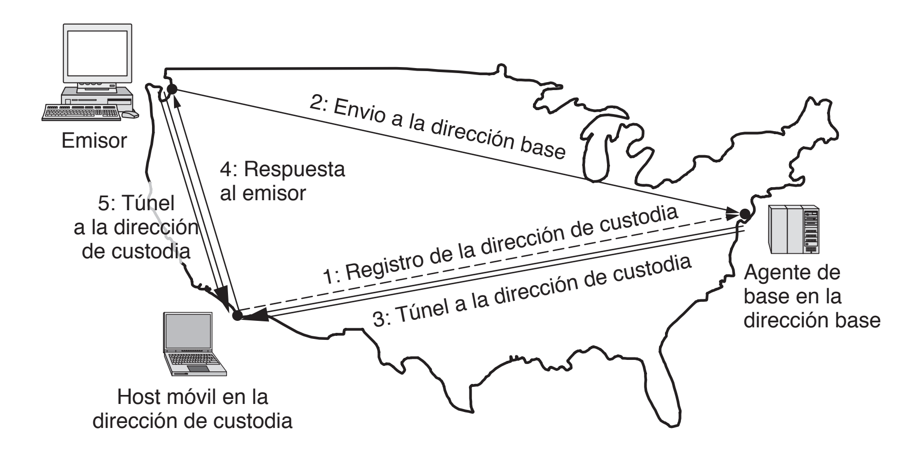


Figura 5-19. Enrutamiento de paquetes para hosts móviles.

cómo se realiza esto en Internet. La dirección local se denomina **dirección de custodia** (*care of address*). Una vez que el host móvil tiene esta dirección, puede indicar a su agente de base en dónde se encuentra en ese momento. Para ello envía un mensaje de registro al agente de base (paso 1) con la dirección de custodia. El mensaje se muestra con una línea punteada en la figura 5-19 para indicar que es un mensaje de control y no de datos.

A continuación, el emisor envía un paquete de datos al host móvil mediante su dirección permanente (paso 2). Este paquete es enrutado por la red hacia la ubicación base del host, ya que es a donde pertenece la dirección base. En Nueva York, el agente de base intercepta este paquete, puesto que el host móvil está alejado de su ubicación base. Después envuelve o **encapsula** el paquete con un nuevo encabezado y lo envía a la dirección de custodia (paso 3). Este mecanismo se denomina **tunelización** (*tunneling*). Puesto que es muy importante en Internet, posteriormente lo analizaremos con más detalle.

Cuando el paquete encapsulado llega a la dirección de custodia, el host móvil lo desenvuelve y recupera el paquete del emisor. Después, el host móvil envía su paquete de respuesta directamente al emisor (paso 4). La ruta general se denomina **enrutamiento triangular**, debido a que puede ser la menos directa si la ubicación remota está alejada de la ubicación base. Como parte del paso 4, el emisor puede memorizar la dirección de custodia actual. Los paquetes subsiguientes se pueden enrutar directamente al host móvil si se tunelizan a la dirección de custodia (paso 5), ignorando por completo la ubicación base. Si se pierde la conectividad por alguna razón mientras el móvil se traslada, siempre se puede usar la dirección base para localizarlo.

Un aspecto importante que hemos omitido de esta descripción es la seguridad. En general, cuando un host o enrutador recibe un mensaje de la forma “A partir de este momento, envíame todo el correo de Estefanía”, podría tener algunas dudas acerca de con quién está hablando y si se trata de una buena idea. La información de seguridad se incluye en los mensajes de manera que se pueda verificar su validez mediante los protocolos criptográficos que estudiaremos en el capítulo 8.

Existen muchas variaciones del enrutamiento móvil. El esquema anterior está modelado con base en la movilidad IPv6, la forma de movilidad que se utiliza en Internet (Johnson y colaboradores, 2004) y como parte de las redes celulares basadas en IP como UMTS. Aquí mostramos al emisor como un nodo estacionario por cuestión de sencillez, pero los diseños permiten que ambos nodos sean hosts móviles. Como alternativa, el host puede formar parte de una red móvil; por ejemplo, una computadora en un avión. Las extensiones del esquema básico soportan redes móviles sin trabajo por parte de los hosts (Devarapalli y colaboradores, 2005).

Algunos esquemas utilizan un agente foráneo (es decir, remoto), algo similar al agente de base pero en la ubicación foránea, o análogo al VLR (Registro de Ubicación de Visitante, del inglés *Visitor Location*

Register) en las redes celulares. Sin embargo, en esquemas más recientes no se necesita el agente foráneo; los hosts móviles actúan como sus propios agentes foráneos. En cualquier caso, el conocimiento de la ubicación temporal del host móvil se limita a un pequeño número de hosts (por ejemplo, el móvil, el agente de base y los emisores) de modo que muchos de los enrutadores en una red extensa no necesiten recalcular rutas.

Para obtener más información sobre el enrutamiento móvil, consulte también a Perkins (1998, 2002) y a Snoeren y Balakrishnan (2000).

5.2.11 Enrutamiento en redes *ad hoc*

Ya vimos cómo realizar el enrutamiento cuando los hosts son móviles pero los enrutadores son fijos. Un caso aún más extremo es cuando los enrutadores mismos son móviles. Entre las posibilidades se encuentran: trabajadores de emergencia en sitio de terremotos, vehículos militares en un campo de batalla, una flota de barcos en el mar o una reunión de personas con computadoras portátiles en un área que no cuenta con 802.11.

En todos estos casos, y en otros, cada nodo se comunica en forma inalámbrica y actúa como host y como enrutador. A las redes de nodos que simplemente están cerca entre sí se les denomina **redes *ad hoc*** o **MANET** (**Redes *Ad hoc* Móviles**, del inglés *Mobile Ad hoc NETworks*). A continuación las examinaremos con brevedad. Para mayor información, consulte a Perkins (2001).

Lo que distingue a las redes *ad hoc* de las redes cableadas es que en las primeras, de repente se eliminó la topología por la ventana. Los nodos pueden ir y venir o aparecer en nuevos lugares en cualquier momento. En una red cableada, si un enrutador tiene una ruta válida a algún destino, esa ruta continúa siendo válida a menos que ocurran fallas, que con suerte son raras. En una red *ad hoc*, la topología podría cambiar todo el tiempo, por lo que la necesidad o la validez de las rutas puede cambiar en cualquier momento, sin previo aviso. No es necesario decir que estas circunstancias hacen del enrutamiento en redes *ad hoc* algo más desafiante que el enrutamiento en sus contrapartes fijas.

Se ha propuesto una gran variedad de algoritmos de enrutamiento para las redes *ad hoc*. No obstante, como se han usado muy poco en la práctica, en comparación con las redes móviles, no está claro cuál de estos protocolos es el más útil. Como ejemplo analizaremos uno de los algoritmos de enrutamiento más populares: **AODV** (**Vector de Distancia *Ad hoc* bajo Demanda**, del inglés *Ad hoc On-demand Distance Vector*) (Perkins y Royer, 1999). Es pariente del algoritmo de vector de distancia que se adaptó para funcionar en un entorno móvil, en el que con frecuencia los nodos tienen un ancho de banda y tiempos de batería limitados. Ahora veamos cómo descubre y mantiene las rutas.

Descubrimiento de ruta

En el AODV, las rutas a un destino se descubren bajo demanda; es decir, sólo cuando alguien desea enviar un paquete a ese destino. Esto ahorra mucho trabajo que se desperdiciaría de otra manera, cuando la topología cambia antes de usar la ruta. En cualquier instante podemos describir la topología de una red *ad hoc* mediante un grafo de nodos conectados. Dos nodos se conectan (es decir, tienen un arco entre ellos en el grafo) si se pueden comunicar de manera directa mediante sus radios. Un modelo básico pero adecuado, que basta para nuestros fines, es que cada nodo se puede comunicar con los demás nodos que se encuentren dentro de su círculo de cobertura. Las redes reales son más complicadas, con edificios, colinas y demás obstáculos que bloquean la comunicación, y nodos para los que *A* está conectado a *B*, pero *B* no está conectado a *A* debido a que *A* tiene un transmisor más poderoso que *B*. Sin embargo, por simplicidad asumiremos que todas las conexiones son simétricas.

Para describir el algoritmo, considere la red *ad hoc* recién formada de la figura 5-20. Suponga que un proceso en el nodo *A* desea enviar un paquete al nodo *I*. El algoritmo AODV mantiene una tabla de vecto-

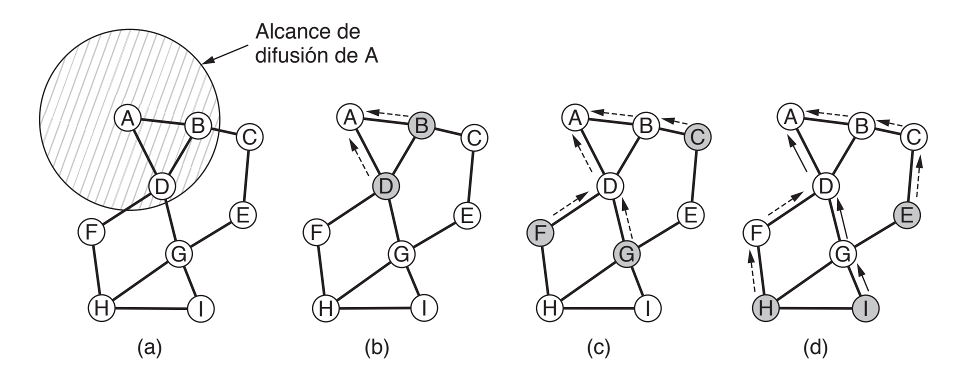


Figura 5-20. a) Alcace de difusión de *A*. (b) Después de que *B* y *D* reciben la difusión de *A*. (c) Después de que *C*, *F* y *G* reciben la difusión de *A*. (d) Después de que *E*, *H* e *I* reciben la difusión de *A*. Los nodos sombreados son nuevos receptores. Las líneas punteadas muestran las posibles rutas invertidas. Las líneas sólidas muestran la ruta descubierta.

res de distancia en cada nodo, codificada por destino, que proporciona información acerca de ese destino, incluyendo a cuál vecino hay que enviar los paquetes con el fin de llegar al destino. Primero, *A* busca en su tabla y no encuentra una entrada para *I*. Ahora tiene que descubrir una ruta a *I*. Debido a esta propiedad de descubrir rutas sólo cuando es necesario, este algoritmo se conoce como "bajo demanda".

Para localizar a *I*, *A* construye un paquete especial de solicitud de ruta (ROUTE REQUEST) y lo difunde usando inundación, como se describío en la sección 5.2.3. La transmisión de *A* llega a *B* y *D*, como se ilustra en la figura 5-20(a). Cada nodo vuelve a difundir la solicitud, que continúa para llegar a los nodos *F*, *G* y *C* en la figura 5-20(c), y a los nodos *H*, *E* e *I* en la figura 5-20(d). Se utiliza un número de secuencia establecido en la fuente para eliminar los duplicados durante la inundación. Por ejemplo, *D* descarta la transmisión proveniente de *B* en la figura 5.20(c) debido a que ya reenvió la solicitud.

En un momento dado la solicitud llega al nodo *I*, que construye un paquete de respuesta de ruta (ROUTE REPLY). Este paquete se envía mediante unidifusión al emisor junto con la ruta invertida, seguida de la solicitud. Para que esto funcione, cada nodo intermedio debe recordar al nodo que le envió la solicitud. Las flechas en la figura 5-20(b)-(d) muestran la información almacenada de la ruta invertida. Cada nodo intermedio también incrementa una cuenta de saltos al reenviar la respuesta. Esto indica a los nodos qué tan alejados están del destino. Las respuestas indican a cada nodo intermedio qué vecino debe usar para llegar al destino: es el nodo que les envió la respuesta. Los nodos intermedios *G* y *D* colocan la mejor ruta que hayan escuchado en sus tablas de enrutamiento, mientras procesan la respuesta. Cuando la respuesta llega a *A*, se ha creado una nueva ruta, *ADGI*.

En una red extensa, el algoritmo genera muchas difusiones, incluso para destinos cercanos. Para reducir la sobrecarga, el alcance de las difusiones se limita mediante el campo *Tiempo de vida del paquete IP*. Este campo es inicializado por el emisor y se decrementa en cada salto. Si llega a 0, el paquete se descarta en vez de difundirlo. Después, el proceso de descubrimiento de ruta se modifica de la siguiente manera. Para localizar un destino, el emisor difunde un paquete de solicitud de ruta (ROUTE REQUEST) con el campo *Tiempo de vida* establecido en 1. Si no regresa una respuesta dentro de un periodo razonable, se envía otro paquete, esta vez con el campo *Tiempo de vida* establecido en 2. Los intentos subsiguientes usan 3, 4, 5, etc. De esta forma la búsqueda se intenta primero de forma local, y después en anillos cada vez más amplios.

Mantenimiento de rutas

Debido a que es posible mover o apagar los nodos, la topología puede cambiar de manera espontánea. Por ejemplo, en la figura 5-20, si *G* se apaga, *A* no se dará cuenta de que la ruta a *I* (*ADGI*) que estaba

utilizando ya no es válida. El algoritmo necesita ser capaz de manejar esto. Cada nodo difunde de manera periódica un mensaje de saludo (*Hello*). Se espera que cada uno de sus vecinos responda a dicho mensaje. Si no se recibe ninguna respuesta, el difusor sabe que el vecino se ha movido del alcance o falló y ya no está conectado a él. De manera similar, si el difusor trata de enviar un paquete a un vecino que no responde, se da cuenta de que el vecino ya no está disponible.

Esta información se utiliza para eliminar rutas que ya no funcionan. Para cada destino posible, cada nodo, *N*, mantiene un registro de sus vecinos activos que le han proporcionado un paquete para ese destino durante los últimos ΔT segundos. Cuando alguno de los vecinos de *N* es inalcanzable, verifica su tabla de enrutamiento para ver cuáles destinos tienen rutas que utilicen el vecino que ya no se puede localizar. Para cada una de esas rutas, se informa a los vecinos activos que su ruta a través de *N* es ahora inválida y debe eliminarse de sus tablas de enrutamiento. En nuestro ejemplo, *D* elimina sus entradas para *G* e *I* de su tabla de enrutamiento y notifica a *A*, quien elimina su entrada para *I*. En el caso general, los vecinos activos avisan a sus vecinos activos y así en lo sucesivo, en forma recursiva hasta que se eliminan de todas las tablas de enrutamiento todas las rutas que dependen del nodo ahora ilocalizable.

En esta etapa, ya se eliminaron las rutas inválidas de la red y los emisores pueden encontrar nuevas rutas válidas mediante el mecanismo de descubrimiento que describimos. Sin embargo, existe una complicación. Recordemos que los protocolos de vector de distancia pueden sufrir de una convergencia lenta o de problemas de conteo al infinito después de un cambio en la topología, en donde pueden confundir las rutas antiguas e inválidas con las rutas nuevas y válidas.

Para asegurar una convergencia rápida, las rutas incluyen un número de secuencia controlado por el destino. El número de secuencia de destino es como un reloj lógico. El destino lo incrementa cada vez que envía un nuevo paquete de respuesta de ruta (ROUTE REPLY). Para preguntar por una ruta nueva, los emisores incluyen en el paquete ROUTE REQUEST el número de secuencia de destino de la última ruta que usaron, que puede ser el número de secuencia de la ruta que se acaba de eliminar, o 0 como un valor inicial. La petición se difundirá hasta encontrar una ruta con un número de secuencia más alto. Los nodos intermedios almacenan las rutas que tienen un número de secuencia más alto, o la menor cantidad de saltos para el número de secuencia actual.

En aras de un protocolo bajo demanda, los nodos intermedios sólo almacenan las rutas en uso. La demás información sobre rutas aprendida durante las difusiones expira después de un retardo corto. Al descubrir y almacenar sólo las rutas que se utilizan es posible ahorrar ancho de banda y vida de la batería, en comparación con un protocolo de vector de distancia estándar, que actualiza las difusiones en forma periódica.

Hasta ahora sólo hemos considerado una sola ruta, de *A* a *I*. Para ahorrar más recursos, se comparte el descubrimiento de rutas y el mantenimiento cuando las rutas se traslapan. Por ejemplo, si *B* también desea enviar paquetes a *I*, llevará a cabo el descubrimiento de rutas. Sin embargo, en este caso la solicitud llegará primero a *D*, que ya tiene una ruta a *I*. El nodo *D* puede entonces generar una respuesta para indicar a *B* la ruta sin requerir ningún trabajo adicional.

Existen muchos más esquemas de enrutamiento ad hoc. Otro esquema bajo demanda muy conocido es DSR (Enrutamiento Dinámico de Origen, del inglés *Dynamic Source Routing*) (Johnson y colaboradores, 2001). El esquema GPSR (Enrutamiento sin Estado con Perímetro Codicioso, del inglés *Greedy Perimeter Stateless Routing*) explora una estrategia diferente, basada en la geografía (Karp y Kung, 2000). Si todos los nodos conocen sus posiciones geográficas, el reenvío a un destino puede proceder sin necesidad de calcular las rutas, con sólo dirigirse en el sentido correcto y regresar en círculos para escapar de cualquier camino sin salida. Los protocolos que sobrevivan dependerán de los tipos de redes ad hoc que demuestren ser útiles en la práctica.

5.3 ALGORITMOS DE CONTROL DE CONGESTIÓN

Cuando hay demasiados paquetes presentes en una red (o en una parte de ella), hay retardo o pérdida en los paquetes y se degrada el desempeño. Esta situación se llama **congestión**. Las capas de red y de transporte comparten la responsabilidad de manejar la congestión. Como ésta ocurre dentro de la red, la capa de red es quien la experimenta en forma directa y en última instancia debe determinar qué hacer con los paquetes sobrantes. Sin embargo, la manera más efectiva de controlar la congestión es reducir la carga que la capa de transporte coloca en la red. Para ello se requiere que las capas de red y de transporte trabajen en conjunto. En este capítulo analizaremos los aspectos de la congestión relacionados con la red. En el capítulo 6 completaremos el tema al cubrir los aspectos de la congestión relacionados con el transporte.

En la figura 5-21 se muestra el comienzo de la congestión. Cuando la cantidad de paquetes que el host envía a la red está muy por debajo de su capacidad de transporte, la cantidad entregada es proporcional a la cantidad enviada. Si se envía el doble de paquetes, se entrega el doble de ellos. Sin embargo, a medida que la carga ofrecida se acerca a la capacidad de transporte, las ráfagas de tráfico llenan ocasionalmente los búferes dentro de los enrutadores y se pierden algunos paquetes. Estos paquetes perdidos consumen una parte de la capacidad, por lo que la cantidad de paquetes entregados cae por debajo de la curva ideal. Ahora la red está congestionada.

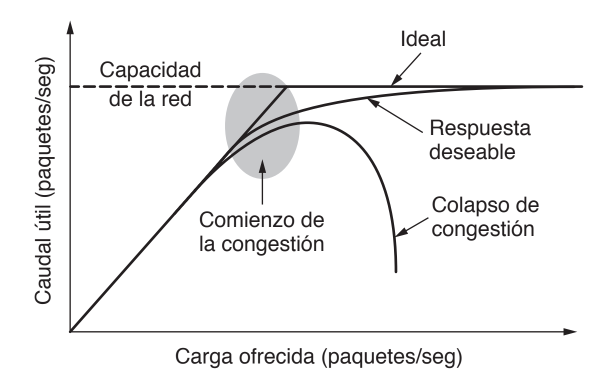


Figura 5-21. Con demasiado tráfico, el desempeño se reduce bruscamente.

A menos que la red esté bien diseñada, puede experimentar un **colapso por congestión**, en donde el desempeño se desploma a medida que aumenta la carga ofrecida más allá de la capacidad. Esto puede ocurrir debido a que los paquetes se pueden retrasar el tiempo suficiente dentro de la red como para que ya no sean útiles cuando salgan de ella. Por ejemplo, en los primeros días de Internet, el tiempo que invertía un paquete en esperar a que se enviaran primero los paquetes acumulados a través de un enlace de 56 kbps, podía llegar al máximo permitido para permanecer en la red. Entonces había que descartarlo. Un modo de falla distinto ocurre cuando los emisores retransmiten los paquetes que están muy atrasados, creyendo que se perdieron. En este caso la red entregará copias del mismo paquete, desperdiciando de nuevo su capacidad. Para capturar estos factores, al eje y de la figura 5-21 se le nombró como **caudal útil** (*goodput*), que es la tasa a la que se entregan los paquetes *útiles* en la red.

Nos gustaría diseñar redes que evitaran la congestión siempre que fuera posible y que no sufrieran de un colapso por congestión en caso de que se atascaran. Por desgracia, no podemos evitar la congestión por completo. Si de repente empiezan a llegar flujos de paquetes en tres o cuatro líneas de entrada y todos necesitan la misma línea de salida, se acumulará una cola. Si no hay suficiente memoria para

almacenarlos a todos, se perderán paquetes. Tal vez agregar memoria ayude hasta cierto punto, pero Nagle (1987) descubrió que si los enrutadores tienen una cantidad infinita de memoria, la congestión empeora en vez de mejorar. Esto se debe a que, para cuando los paquetes llegan a la parte frontal de la cola, ya expiraron (repetidas veces) y se enviaron duplicados. Esto empeora las cosas, no las mejora: conduce al colapso por congestión.

Los enlaces o enrutadores con poco ancho de banda, que procesan paquetes con más lentitud que la tasa de transmisión de la línea, también se pueden congestionar. En este caso, se puede mejorar la situación al dirigir parte del tráfico de manera que se aleje del embotellamiento y se vaya a otras partes de la red. Sin embargo, en un momento dado todas las regiones de la red estarán congestionadas. En esta situación, no hay otra alternativa más que deshacerse de una parte de la carga o construir una red más rápida.

Vale la pena recalcar la diferencia entre el control de la congestión y el control de flujo, pues la relación es muy sutil. El control de congestión se ocupa de asegurar que la red sea capaz de transportar el tráfico ofrecido. Es un asunto global, en el que interviene el comportamiento de todos los hosts y enrutadores. En contraste, el control de flujo se relaciona con el tráfico entre un emisor particular y un receptor particular. Su tarea es asegurar que un emisor rápido no pueda transmitir datos de manera continua a una velocidad mayor que la que puede absorber los paquetes el receptor.

Para ver la diferencia entre estos dos conceptos, considere una red compuesta de enlaces de fibra óptica de 100 Gbps en la que una supercomputadora está tratando de transferir por la fuerza un archivo extenso a una computadora personal que sólo puede manejar 1 Gbps. Aunque no hay congestión (la red misma no es el problema), se requiere control de flujo para obligar a la supercomputadora a detenerse con frecuencia para darle a la computadora personal un momento de respiro.

En el otro extremo, considere una red con líneas de 1 Mbps y 1000 computadoras grandes, la mitad de las cuales trata de transferir archivos a 100 kbps a la otra mitad. Aquí el problema no es que los emisores rápidos saturen a los receptores lentos, sino que el tráfico ofrecido total excede lo que la red puede manejar.

La razón por la que se confunden con frecuencia el control de congestión y el control de flujo es que la mejor manera de lidiar con ambos problemas es hacer que el host reduzca su velocidad. Así, un host puede recibir un mensaje de “reducción de velocidad” porque el receptor no puede manejar la carga o porque la red no la puede manejar. En el capítulo 6 regresaremos a este punto.

Comenzaremos nuestro estudio del control de congestión mediante un análisis de los métodos que se pueden usar en distintas escalas de tiempo. Después analizaremos los métodos para prevenir que ocurra la congestión en primer lugar, seguidos de los métodos para lidiar con la congestión una vez que se ha establecido.

5.3.1 Métodos para el control de la congestión

La presencia de congestión significa que la carga es (temporalmente) mayor de la que los recursos (en una parte de la red) pueden manejar. Dos soluciones vienen a la mente: aumentar los recursos o reducir la carga. Como se muestra en la figura 5-22, estas soluciones por lo general se aplican en distintas escalas de tiempo, para prevenir la congestión o reaccionar ante ella una vez que se presenta.

La manera más básica de evitar la congestión es construir una red que coincida bien con el tráfico que transmita. Si hay un enlace con poco ancho de banda en la ruta a través de la cual se dirige la mayor parte del tráfico, es probable que haya congestión. Algunas veces se pueden agregar recursos en forma dinámica cuando hay un problema grave de congestión; por ejemplo, activar enrutadores de repuesto o habilitar líneas que por lo general se usan sólo como respaldos (para que el sistema sea tolerante a fallas), o comprar ancho de banda en el mercado abierto. Lo más frecuente es que los enlaces y enrutadores que se utilizan

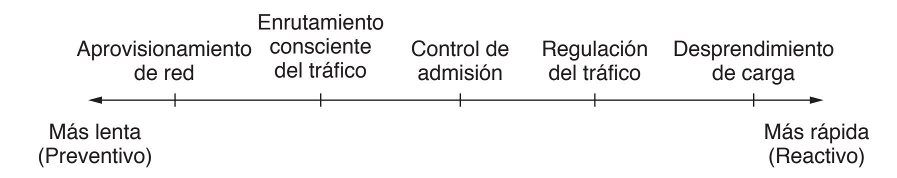


Figura 5-22. Escalas de tiempo de los métodos para el control de la congestión.

mucho en forma regular se actualicen a la primera oportunidad. A esto se le conoce como **aprovisionamiento** y ocurre en una escala de tiempo de meses, con base en las tendencias de tráfico de largo plazo.

Para aprovechar al máximo la capacidad existente de la red, las rutas se pueden ajustar a los patrones de tráfico que cambian durante el día, a medida que los usuarios de la red entran y salen en distintas zonas horarias. Por ejemplo, se pueden cambiar las rutas para desviar el tráfico de las mismas que se utilizan mucho si se cambian las ponderaciones de las rutas más cortas. Ciertas estaciones de radio locales tienen helicópteros que vuelan por sus ciudades para informar sobre la congestión en las carreteras, de modo que sus radioescuchas móviles puedan enrutar sus paquetes (autos) alrededor de los sitios con mucho tráfico. A esto se le conoce como **enrutamiento consciente del tráfico** (*traffic-aware routing*). También es útil dividir el tráfico entre varias rutas.

Sin embargo, en ocasiones no es posible aumentar la capacidad. La única forma entonces para combatir la congestión es reducir la carga. En una red de circuitos virtuales, se podrían rechazar las nuevas conexiones si provocaran el congestionamiento de la red. A esto se le conoce como **control de admisión**.

A un nivel más detallado, cuando la congestión es inminente la red puede distribuir retroalimentación a las fuentes cuyos flujos de tráfico sean responsables del problema. La red puede solicitar que estas fuentes regulen su tráfico, o puede reducir el tráfico por sí misma.

Dos dificultades con este método son: cómo identificar el comienzo de la congestión y cómo informar a la fuente que necesita reducir su velocidad. Para lidiar con la primera cuestión, los enrutadores pueden monitorear la carga promedio, el retardo de encolamiento o la pérdida de paquetes. En todos los casos, los números crecientes indican un aumento en la congestión.

Para lidiar con la segunda cuestión, los enrutadores deben participar en un lazo de retroalimentación con las fuentes. Para que un esquema funcione de manera correcta, es necesario ajustar la escala de tiempo con cuidado. Si un enrutador grita ALTO cada vez que llegan dos paquetes seguidos y grita SIGA cada vez que está inactivo por 20 μseg, el sistema oscilará sin control y nunca convergerá. Por otra parte, si espera 30 minutos para asegurarse antes de decir algo, el mecanismo de control de congestión reaccionará tan despacio que no será de utilidad. Encontrar una retroalimentación oportuna no es un asunto trivial. Una cuestión adicional es tener enrutadores que envíen más mensajes cuando la red ya se encuentra congestionada.

Por último, cuando todo lo demás falla, la red se ve obligada a descartar los paquetes que no puede entregar. El nombre general para esto es **desprendimiento de carga** (*load shedding*). Una buena política para elegir qué paquetes descartar puede ayudar a evitar un colapso por congestión.

5.3.2 Enrutamiento consciente del tráfico

El primer método que examinaremos es el enrutamiento consciente del tráfico. Los esquemas de enrutamiento que vimos en la sección 5.2 utilizaban ponderaciones de enlaces fijas. Estos esquemas se adaptaban a los cambios en la topología, pero no a los cambios en la carga. El objetivo al tener en cuenta la carga al calcular las rutas es desviar el tráfico de los puntos más activos que serán los primeros lugares en la red en experimentar congestión.

La manera más directa de hacer esto es establecer la ponderación de enlaces de manera que sea una función del ancho de banda del enlace (fijo) y el retardo de propagación más la carga medida (variable) o el retardo de encolamiento promedio. Así, las rutas de menor ponderación favorecerán a las rutas que tengan cargas más ligeras, siendo todo lo demás igual.

El enrutamiento consciente del tráfico se utilizó en los primeros días de Internet de acuerdo con este modelo (Khanna y Zinky, 1989). Sin embargo, existe un riesgo. Considere la red de la figura 5-23, que está dividida en dos partes, Este y Oeste, conectada mediante dos enlaces, *CF* y *EI*. Suponga que la mayor parte del tráfico entre Este y Oeste utiliza el enlace *CF* y, como resultado, este enlace está muy cargado con retardos muy grandes. Si se incluye el retardo de encolamiento en la ponderación utilizada para el cálculo de la ruta más corta, el enlace *EI* será más atractivo. Una vez que se hayan instalado las nuevas tablas de enrutamiento, la mayor parte del tráfico Este-Oeste viajará ahora a través de *EI*, y se cargará este enlace. Como consecuencia, en la siguiente actualización *CF* parecerá ser la ruta más corta. De esta forma, las tablas de enrutamiento pueden oscilar mucho, lo cual conducirá a un enrutamiento errático y muchos problemas potenciales.

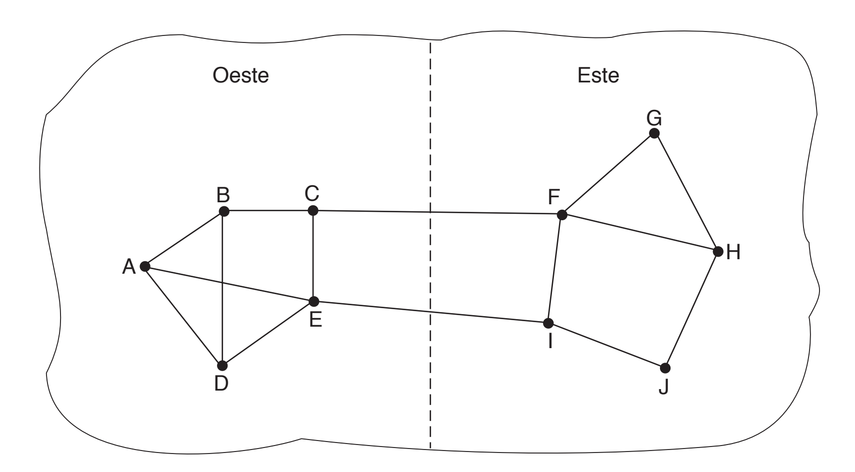


Figura 5-23. Una red en la cual las partes Este y Oeste se conectan mediante dos enlaces.

Si se ignora la carga y sólo se consideran el ancho de banda y el retardo de propagación, este problema no ocurre. Los intentos de incluir la carga pero cambiar las ponderaciones dentro de un rango estrecho sólo reducen las oscilaciones de enrutamiento. Hay dos técnicas que pueden contribuir a una solución exitosa. La primera es el enrutamiento multitrayectoria, en donde puede haber múltiples rutas de un origen a un destino. En nuestro ejemplo, esto significa que el tráfico se puede esparcir a través de ambos enlaces de Este a Oeste. La segunda es que el esquema de enrutamiento desvíe el tráfico a través de las rutas con la suficiente lentitud como para que pueda converger, como en el esquema de Gallagher (1977).

Dadas estas dificultades, en Internet los protocolos de enrutamiento por lo general no ajustan sus rutas dependiendo de la carga, sino que los ajustes se realizan fuera del protocolo de enrutamiento, al cambiar lentamente sus entradas. A esto se le denomina **ingeniería de tráfico**.

5.3.3 Control de admisión

Una técnica que se utiliza mucho en las redes de circuitos virtuales para la congestión a raya es el **control de admisión**. La idea es simple: no se debe establecer un nuevo circuito virtual a menos que la red pueda transportar el tráfico adicional sin congestionarse. Por lo tanto, pueden fallar los intentos por establecer un

circuito virtual. Esto es mejor que la alternativa, ya que dejar entrar más personas cuando la red está ocupada sólo empeora las cosas. Por analogía, en el sistema telefónico, cuando un conmutador se sobrecarga, practica el control de admisión al no dar tonos de marcado.

El truco con este método es averiguar cuándo puede un nuevo circuito virtual provocar una congestión. La tarea es simple en el sistema telefónico debido al ancho de banda fijo de las llamadas (64 kbps para audio no comprimido). Sin embargo, los circuitos virtuales en las redes de computadoras vienen en todas las formas y tamaños. Por ende, el circuito debe incluir alguna caracterización de su tráfico si se va a aplicar el control de admisión.

A menudo el tráfico se describe en términos de su tasa de transmisión y forma. El problema de cómo describirlo en una forma simple pero significativa es difícil, ya que por lo general el tráfico es de ráfagas; la tasa promedio es sólo la mitad de la historia. Por ejemplo, el tráfico que varía mientras se navega por la web es más difícil de manejar que una película de flujo continuo con la misma velocidad real de transporte a largo plazo, pues es más probable que las ráfagas del tráfico web congestionen los enrutadores en la red. Un descriptor de uso común que captura este efecto es la **cubeta con goteo** (*leaky bucket*) o **cubeta con token** (*token bucket*). Una cubeta con goteo tiene dos parámetros que vinculan la tasa promedio y el tamaño de la ráfaga instantánea de tráfico. Como las cubetas con goteo se utilizan mucho para la calidad del servicio, las analizaremos con detalle en la sección 5.4.

Armada con las descripciones del tráfico, la red puede decidir si admite o no el nuevo circuito virtual. Una posibilidad es que la red reserve suficiente capacidad a lo largo de las rutas de cada uno de sus circuitos virtuales, de modo que no ocurra una congestión. En este caso, la descripción del tráfico es un acuerdo de servicio en cuanto a lo que la red garantizará a sus usuarios. Hemos prevenido la congestión, pero viramos hacia el tema relacionado de la calidad del servicio un poco antes de tiempo; regresaremos a este tema en la siguiente sección.

Aun sin hacer garantías, la red puede usar las descripciones del tráfico para el control de admisión. De esta forma, la tarea es estimar cuántos circuitos caben dentro de la capacidad de transporte de la red sin que haya congestión. Suponga que los circuitos virtuales que pueden enviar tráfico a tasas de hasta 10 Mbps pasan por el mismo enlace físico de 100 Mbps. ¿Cuántos circuitos se deben admitir? Sin duda, se pueden admitir 10 sin arriesgarse a una congestión, pero esto es un desperdicio en el caso normal puesto que sería muy raro que los 10 circuitos transmitieran a toda velocidad al mismo tiempo. En las redes reales, las mediciones del comportamiento en el pasado que capturan las estadísticas de las transmisiones, se pueden usar para estimar el número de circuitos que se pueden admitir, para intercambiar un mejor desempeño por un riesgo aceptable.

El control de admisión también se puede combinar con el enrutamiento consciente del tráfico si se consideran las rutas alrededor de los puntos con mayor tráfico como parte del procedimiento de establecimiento. Por ejemplo, considere la red que se muestra en la figura 5-24(a), en donde hay dos enrutadores congestionados según se indica.

Suponga que un host conectado al enrutador *A* desea establecer una conexión a un host conectado al enrutador *B*. Por lo general, esta conexión pasaría a través de uno de los enrutadores congestionados. Para evitar esta situación, podemos redibujar la red como se muestra en la figura 5-24(b), en donde se omiten los enrutadores congestionados y todas sus líneas. La línea punteada muestra una posible ruta para el circuito virtual que evita a los enrutadores congestionados. Shaikh y colaboradores (1999) proporcionan un diseño para este tipo de enrutamiento sensible a la carga.

5.3.4 Regulación de tráfico

En Internet y en muchas otras redes de computadoras, los emisores ajustan sus transmisiones para enviar tanto tráfico como la red pueda distribuir. En este escenario, la red aspira a operar justo antes de que

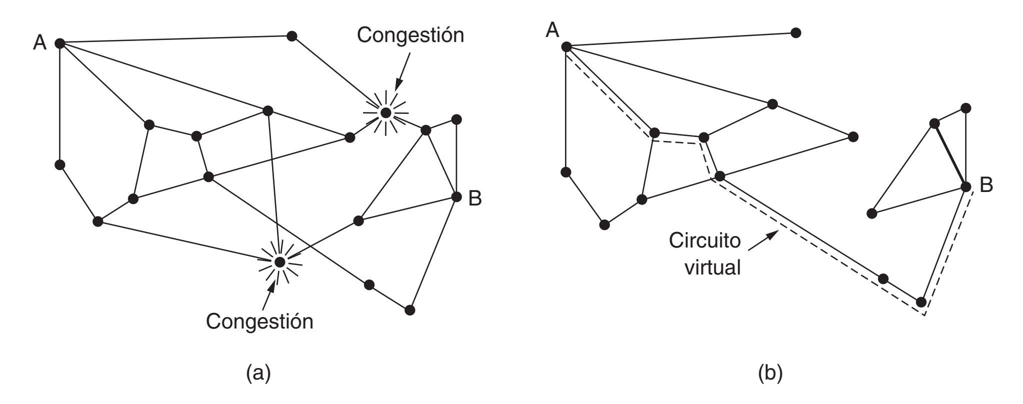


Figura 5-24. (a) Una red congestionada. (b) La parte de la red que no está congestionada. También se muestra un circuito virtual de *A* a *B*.

comience la congestión. Cuando la congestión es inminente, debe pedir a los emisores que reduzcan sus transmisiones y su velocidad. Esta retroalimentación es algo común, en vez de una situación excepcional. En ocasiones se utiliza el término **evasión de congestión** para contrastar este punto de operación con el punto en que la red se ha congestionado (demasiado).

Ahora veamos algunos métodos para regular el tráfico que se pueden usar en las redes de datagramas y en las de circuitos virtuales. Cada método debe resolver dos problemas. En primer lugar, los enrutadores deben determinar cuando se acerca la congestión, siendo lo ideal antes de que haya llegado. Para ello, cada enrutador puede monitorear en forma continua los recursos que utiliza. Tres posibilidades son: usar los enlaces de salida, el búfer para los paquetes puestos en cola dentro del enrutador y el número de paquetes que se pierden debido a una capacidad insuficiente. De estas posibilidades, la segunda es la más útil. Los promedios de uso no justifican directamente las ráfagas de la mayoría del tráfico; un uso del 50% puede ser bajo para un tráfico uniforme y demasiado alto para un tráfico muy variable. Las cuentas de los paquetes perdidos llegan demasiado tarde. La congestión ya ha comenzado para cuando se empiezan a perder los paquetes.

El retardo de encolamiento dentro de los enrutadores captura de manera directa cualquier congestión experimentada por los paquetes. Debe ser bajo la mayor parte del tiempo, pero se disparará cuando haya una ráfaga de tráfico que genere una acumulación de paquetes. Para mantener una buena estimación del retardo de encolamiento, d, se puede realizar un muestreo periódico de la longitud de cola instantánea, s, y se puede actualizar d de acuerdo con

$$d_{\text{nueva}} = \alpha d_{\text{anterior}} + (1 - \alpha)s$$

en donde la constante α determina qué tan rápido olvida el enrutador el historial reciente. A esto se le llama **EWMA** (**Promedio Móvil Ponderado Exponencialmente**, del inglés *Exponentially Weighted Moving Average*). Este promedio corrige las fluctuaciones y es equivalente a un filtro pasabajas. Cada vez que d se mueve por encima del umbral, el enrutador detecta el comienzo de la congestión.

El segundo problema es que los enrutadores deben entregar una retroalimentación oportuna a los emisores que provocan la congestión. Ésta se experimenta en la red, pero para aliviarla se requiere una acción de parte de los emisores que están usando la red. Para entregar la retroalimentación, el enrutador debe identificar a los emisores apropiados. Después, debe advertirles con cuidado, sin enviar más paquetes a la red que ya está congestionada. Los distintos esquemas usan mecanismos de retroalimentación diferentes, como veremos a continuación.

Paquetes reguladores

La manera más directa de notificar a un emisor sobre la congestión es decírselo directamente. En este método, el enrutador selecciona un paquete congestionado y envía un **paquete regulador** de vuelta al host de origen, proporcionándole el destino encontrado en el paquete. El paquete original se puede etiquetar (se activa un bit de encabezado) de modo que no genere más paquetes reguladores más adelante en la ruta y después se reenvía de la manera usual. Para evitar aumentar la carga en la red durante un momento de congestión, el enrutador tal vez sólo envíe paquetes reguladores a una tasa de transmisión baja.

Cuando el host de origen obtiene el paquete regulador, se le pide que reduzca el tráfico enviado al destino especificado; por ejemplo, un 50%. En una red de datagramas, con sólo elegir paquetes al azar cuando hay congestión es probable que los paquetes reguladores se envíen a los emisores rápidos, ya que tendrán la mayor parte de los paquetes en la cola. La retroalimentación implícita en este protocolo puede ayudar a evitar la congestión sin necesidad de regular a ninguno de los emisores, a menos que ocasione problemas. Por la misma razón, es probable que se envíen varios paquetes reguladores a un host y destino específicos. El host debe ignorar estos paquetes reguladores adicionales durante el intervalo fijo, hasta que tenga efecto su reducción del tráfico. Después de ese periodo, los demás paquetes reguladores indican que la red sigue estando congestionada.

Un ejemplo de un paquete regulador utilizado en los inicios de Internet es el mensaje SOURCE-QUENCH (Postel, 1981). Aunque nunca ganó popularidad, en parte debido a que las circunstancias en las que se generó y el efecto que tuvo no se especificaron con claridad. La Internet moderna usa un diseño de notificación alternativo que describiremos a continuación.

Notificación explícita de congestión

En vez de generar paquetes adicionales para advertir sobre la congestión, un enrutador puede etiquetar cualquier paquete que reenvíe (para lo cual establece un bit en el encabezado de éste) para indicar que está experimentando una congestión. Cuando la red entrega el paquete, el destino puede observar que hay congestión e informa al emisor sobre ello cuando envíe un paquete de respuesta. En consecuencia, el emisor puede regular sus transmisiones como antes.

A este diseño se le conoce como **ECN** (**Notificación Explícita de Congestión**, del inglés *Explicit Congestion Notification*) y se utiliza en Internet (Ramakrishnan y colaboradores, 2001). Es un refinamiento de los primeros protocolos de señalización de congestión, de los cuales el más notable fue el esquema de retroalimentación binaria de Ramakrishnan y Jain (1988) que se utilizó en la arquitectura de DECNET. Se utilizan dos bits en el encabezado del paquete IP para registrar el momento en que el paquete experimenta una congestión. Los paquetes no están marcados cuando se envían, como se ilustra en la figura 5-25. Si alguno de los enrutadores por los que pasen está congestionado, entonces ese enrutador marcará el paquete para indicar que experimentó una congestión mientras lo reenvía. Después, el destino repetirá cualquier marca de vuelta al emisor, como una señal de congestión explícita en su siguiente pa-

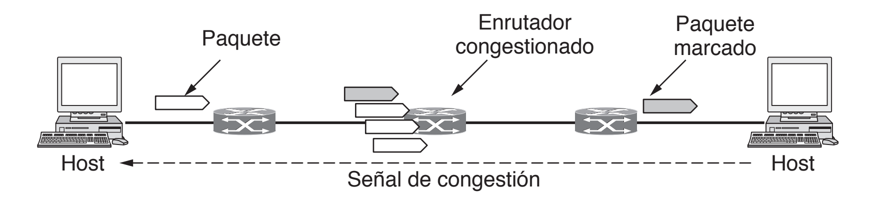


Figura 5-25. Notificación explícita de congestión.

quete de respuesta. Esto se muestra con una línea punteada en la figura para indicar que ocurre por encima del nivel de IP (por ejemplo, en TCP). El emisor debe entonces regular sus transmisiones, como en el caso de los paquetes reguladores.

Contrapresión de salto por salto

A largas distancias o altas velocidades, muchos paquetes nuevos se pueden transmitir una vez que se haya señalado la congestión debido al retardo antes de que la señal haga efecto. Por ejemplo, considere a un host en San Francisco (enrutador *A* de la figura 5-26) que envía tráfico a un host en Nueva York (enrutador *D* de la figura 5-26) a una velocidad OC-3 de 155 Mbps. Si el host de Nueva York empieza a quedarse sin búferes, un paquete regulador tardará alrededor de 40 mseg en regresar a San Francisco e indicarle que reduzca su velocidad. Una indicación ECN tardará aún más, ya que se entrega a través del destino. La propagación de paquetes reguladores se ilustra como los pasos segundo, tercero y cuarto en la figura 5-26(a). En esos 40 mseg, se habrán enviado otros 6.2 megabits. Incluso si el host en San Francisco se desconecta de inmediato, los 6.2 megabits en la línea seguirán fluyendo y habrá que lidiar con ellos. Sólo hasta el séptimo diagrama en la figura 5-26(a) será cuando el enrutador de Nueva York observe un flujo más lento.

Un método alternativo es hacer que el paquete regulador ejerza su efecto en cada salto por el que pase, como se muestra en la secuencia de la figura 5-26(b). Aquí, tan pronto como el paquete regulador llega a *F*, se obliga a *F* a reducir el flujo que va hacia *D*. Para hacer esto, *F* tendría que destinar más búferes a la conexión, ya que la fuente aún está transmitiendo a toda velocidad, pero da a *D* un alivio inmediato, como un remedio contra el dolor de cabeza en un comercial de televisión. En el siguiente paso, el paquete regulador llega a *E*, e indica a éste que reduzca el flujo a *F*. Esta acción impone una mayor demanda sobre los búferes de *E* pero da un alivio inmediato a *F*. Por último, el paquete regulador llega a *A* y el flujo se reduce.

El efecto neto de este esquema de salto por salto es proporcionar un alivio rápido en el punto de congestión, a expensas de usar más búferes ascendentes. De esta manera se puede cortar la congestión de raíz sin que se pierdan paquetes. La idea se estudia con mayor detalle en Mishra y colaboradores (1996).

5.3.5 Desprendimiento de carga

Cuando ninguno de los métodos anteriores elimina la congestión, los enrutadores pueden sacar la artillería pesada: el **desprendimiento de carga**, que es una manera rebuscada de decir que cuando se inunda a los enrutadores con paquetes que no pueden manejar, simplemente se tiran. El término viene del mundo de la generación de energía eléctrica, donde se refiere a la práctica de instalaciones que producen apagones intencionales en ciertas áreas para salvar a la red completa de colapsarse en días calurosos de verano, en los que la demanda de energía eléctrica excede por mucho el suministro.

La pregunta clave para un enrutador abrumado por paquetes es cuáles paquetes tirar. La opción preferida puede depender del tipo de aplicaciones que utiliza la red. En una transferencia de archivos vale más un paquete viejo que uno nuevo, puesto que si el enrutador se deshace del paquete 6 y mantiene los paquetes 7 a 10, por ejemplo, sólo obligará al receptor a esforzarse más por colocar en el búfer los datos que no puede usar todavía. En contraste, para los medios en tiempo real, un paquete nuevo vale más que uno viejo. Esto se debe a que los paquetes se vuelven inútiles si se retardan y se pasa el tiempo en el que se deben reproducir para el usuario.

A la primera política (más viejo es mejor que más nuevo) se le conoce comúnmente como **vino** (*wine*) y a la segunda (más nuevo es mejor que más viejo) con frecuencia se le llama **leche** (*milk*), ya que la mayoría de las personas preferirían beber leche nueva y por lo contrario vino viejo.

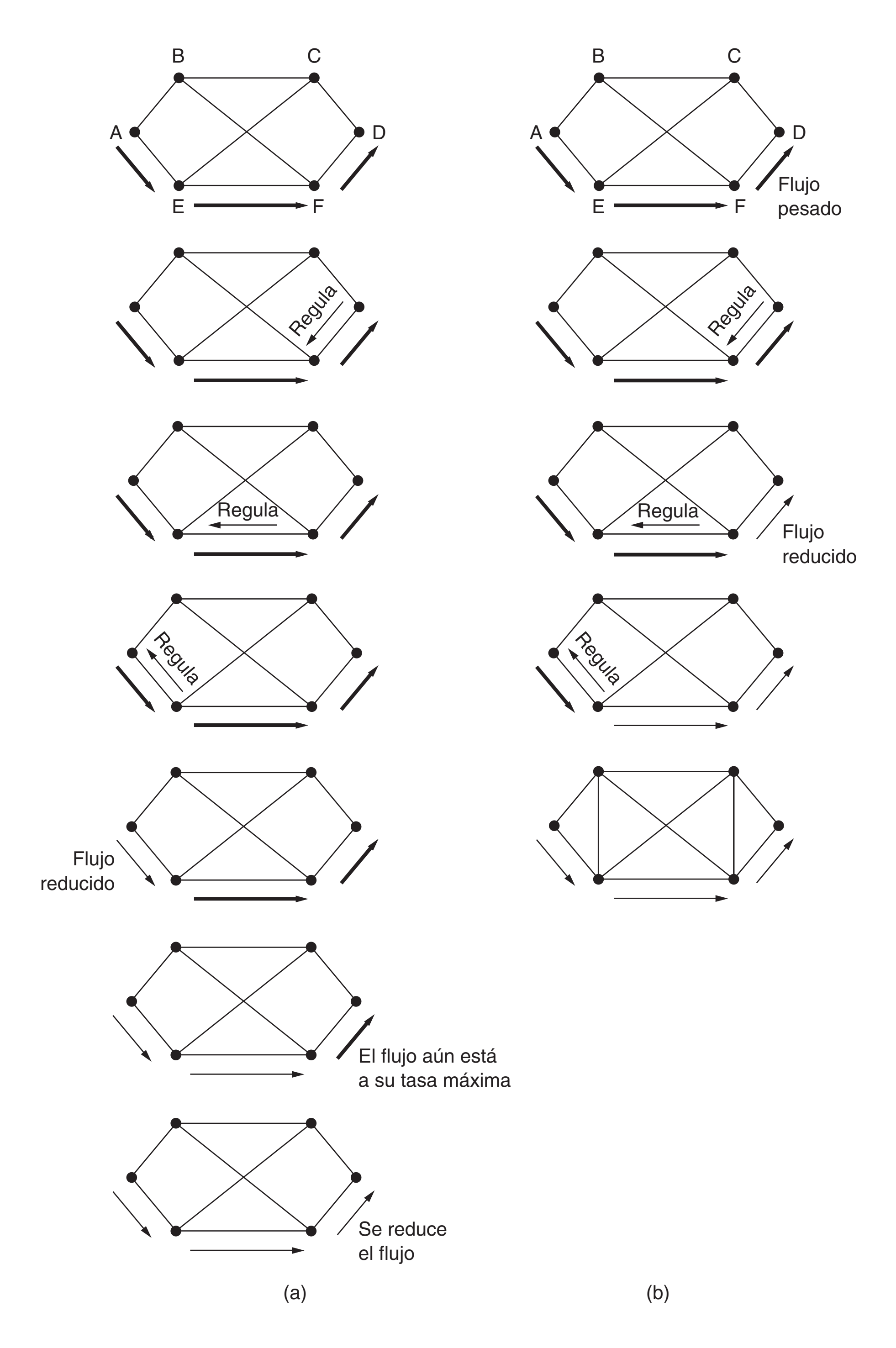


Figura 5-26. (a) Un paquete regulador que afecta sólo al origen. (b) Un paquete regulador que afecta a cada salto por el que pasa.

Un desprendimiento de carga más inteligente requiere la cooperación de los emisores. Un ejemplo es el de los paquetes que transmiten información de enrutamiento. Estos paquetes son más importantes que los paquetes de datos regulares, ya que establecen rutas; si se extravían, la red puede perder conectividad. Otro ejemplo es el de los algoritmos para comprimir video, como MPEG, que transmiten en forma periódica toda una trama y después envían tramas como diferencias respecto a la última trama completa. En este caso es preferible desprenderse de un paquete que forma parte de una diferencia, que desprenderse de uno que forme parte de una trama completa, ya que los futuros paquetes dependerán de la trama completa.

Para implementar una política inteligente de descarte, las aplicaciones deben marcar sus paquetes para indicar a la red qué tan importantes son. Así, al tener que descartar paquetes, los enrutadores pueden eliminar primero los paquetes de la clase menos importante, luego los de la siguiente clase más importante, y así en lo sucesivo.

Por supuesto, a menos que haya una razón poderosa para evitar marcar todos los paquetes como MUY IMPORTANTE–NUNCA, NUNCA DESCARTAR, nadie lo hará. A menudo se usan la contabilidad y el dinero para disuadir una marcación frívola. Por ejemplo, la red podría permitir a los emisores enviar con más rapidez de la permitida por el servicio que compraron si marcan los paquetes en exceso como de baja prioridad. En realidad, dicha estrategia no es una mala idea, ya que hace un uso más eficiente de los recursos inactivos al permitir que los hosts los utilicen mientras que nadie más esté interesado, pero sin establecer un derecho sobre ellos cuando los tiempos se vuelven difíciles.

Detección temprana aleatoria

Es más efectivo lidiar con la congestión cuando apenas empieza que dejar que dañe la red y luego tratar de solucionarlo. Esta observación conduce a un giro interesante sobre el desprendimiento de carga: descartar paquetes antes de que se agote realmente el espacio en el búfer.

La motivación para esta idea es que la mayoría de los hosts de Internet todavía no obtienen señales de congestión de los enrutadores en la forma de ECN; la única indicación confiable de congestión que reciben los hosts de la red es la pérdida de paquetes. Después de todo, es difícil construir un enrutador que no descarte paquetes cuando esté sobrecargado. Por ende, los protocolos de transporte (como TCP) están predeterminados a reaccionar a la pérdida como congestión, y en respuesta reducen la velocidad de la fuente. El razonamiento detrás de esta lógica es que TCP se diseñó para redes cableadas y éstas son muy confiables, así que los paquetes que se pierden se deben principalmente a desbordamientos del búfer y no a errores de transmisión. Los enlaces inalámbricos deben recuperar los errores de transmisión en la capa de enlace (de modo que no se vean en la capa de red) para funcionar bien con TCP.

Podemos explotar esta situación para ayudar a reducir la congestión. Al hacer que los enrutadores descarten los paquetes antes de que la situación se vuelva imposible, hay tiempo para que la fuente tome acción antes de que sea demasiado tarde. Un algoritmo popular para realizar esto se conoce como **RED** (**Detección Temprana Aleatoria**, del inglés *Random Early Detection*) (Floyd y Jacobson, 1993). Para determinar cuándo hay que empezar a descartar paquetes, los enrutadores mantienen un promedio acumulado de sus longitudes de cola. Cuando la longitud de cola promedio en algún enlace sobrepasa un umbral, se dice que el enlace está congestionado y se descarta una pequeña fracción de los paquetes al azar. Al elegir los paquetes al azar es más probable que los emisores más rápidos vean un desprendimiento de paquetes; ésta es la mejor opción, ya que el enrutador no puede saber cuál fuente está causando más problemas en una red de datagramas. El emisor afectado observará la pérdida cuando no haya confirmación de recepción, y entonces el protocolo de transporte reducirá su velocidad. Así, el paquete perdido entrega el mismo mensaje que un paquete regulador, pero de manera implícita sin que el enrutador envíe ninguna señal explícita.

Los enrutadores RED mejoran el desempeño en comparación con los enrutadores que sólo descartan paquetes cuando sus búferes están llenos, aunque tal vez requieran de un ajuste para funcionar bien. Por ejemplo, el número ideal de paquetes a descartar depende de cuántos emisores necesiten ser notificados sobre la congestión. Sin embargo, ECN es la opción preferida si está disponible. Funciona exactamente igual, sólo que entrega una señal de congestión de manera explícita en vez de hacerlo como una pérdida; RED se utiliza cuando los hosts no pueden recibir señales explícitas.

5.4 CALIDAD DEL SERVICIO

Las técnicas que analizamos en las secciones anteriores se diseñaron para reducir la congestión y mejorar el rendimiento de la red. Sin embargo, existen aplicaciones (y clientes) que exigen a la red garantías más sólidas de desempeño que "lo mejor que se pueda hacer en base a las circunstancias". En especial, las aplicaciones multimedia necesitan con frecuencia una tasa de transferencia real mínima y una latencia máxima para trabajar. En esta sección continuaremos con nuestro estudio del desempeño de la red, sólo que ahora con un enfoque más pronunciado en las formas de proporcionar una calidad de servicio que coincida con las necesidades de la aplicación. Ésta es un área en la que Internet pasa por un proceso continuo de actualización.

Una solución sencilla para proporcionar una buena calidad del servicio es construir una red con la suficiente capacidad para cualquier tráfico que maneje. El nombre de esta solución es **exceso de aprovisionamiento** (*overprovisioning*). La red resultante transportará el tráfico de la aplicación sin pérdidas considerables y, suponiendo que hay un esquema de enrutamiento decente, entregará los paquetes con una latencia baja. El desempeño no puede ser mejor que esto. En cierto grado, el sistema telefónico tiene un exceso de aprovisionamiento, ya que es raro levantar un teléfono y no obtener un tono de marcado de inmediato. Simplemente hay tanta capacidad disponible, que casi siempre se puede satisfacer la demanda.

El problema con esta solución es que tiene un costo elevado. Básicamente el problema se resuelve con dinero. Los mecanismos de calidad del servicio permiten que una red con menos capacidad cumpla con los requerimientos de la aplicación con la misma eficiencia, a un menor costo. Además, el exceso de aprovisionamiento se basa en el tráfico esperado. No se puede asegurar nada si el patrón de tráfico cambia demasiado. Con los mecanismos de calidad del servicio, la red puede honrar las garantías de desempeño que hace incluso cuando hay picos de tráfico, a costa de rechazar algunas solicitudes.

Es necesario considerar cuatro aspectos para asegurar la calidad del servicio:

1. Lo que las aplicaciones necesitan de la red.
2. Cómo regular el tráfico que entra a la red.
3. Cómo reservar recursos en los enrutadores para garantizar el desempeño.
4. Si la red puede aceptar o no más tráfico en forma segura.

Ninguna técnica maneja todos estos aspectos en forma eficiente. Sin embargo, se ha desarrollado una variedad de técnicas para usarlas en la capa de red (y de transporte). Las soluciones prácticas de calidad del servicio combinan varias técnicas. Para este fin, describiremos dos versiones de calidad del servicio para Internet, conocidas como Servicios integrados y Servicios diferenciados.

5.4.1 Requerimientos de la aplicación

A un conjunto de paquetes que van de un origen a un destino se le denomina **flujo** (Clark, 1988). En una red orientada a conexión, un flujo podría estar constituido por todos los paquetes de una conexión; o en

una red sin conexión, un fujo serían todos los paquetes enviados de un proceso a otro. Podemos caracterizar las necesidades de cada flujo mediante cuatro parámetros principales: ancho de banda, retardo, variación del retardo (*jitter*) y pérdida. En conjunto, estos parámetros determinan la **QoS** (**Calidad del Servicio**, del inglés *Quality of Service*) que requiere el flujo.

En la figura 5-27 se listan varias aplicaciones comunes y el nivel de sus requerimientos de red. Observe que los requerimientos de red son menos exigentes que los requerimientos de aplicación en los casos en que esta última puede mejorar con base en el servicio proporcionado por la red. En particular, las redes no necesitan tener cero pérdidas para una transferencia de archivos confiable, y no necesitan entregar paquetes con retardos idénticos para reproducir audio y video. Cierta cantidad de pérdida se puede reparar con las retransmisiones, y cierta cantidad de variación del retardo se puede solucionar si se colocan paquetes en el búfer del receptor. Sin embargo, no hay nada que las aplicaciones puedan hacer para remediar la situación si la red proporciona muy poco ancho de banda o demasiado retardo.

Aplicación	Ancho de banda	Retardo	Variación del retardo	Pérdida
Correo electrónico.	Bajo	Bajo	Baja	Media
Compartir archivos.	Alto	Bajo	Baja	Media
Acceso a Web.	Medio	Medio	Baja	Media
Inicio de sesión remoto.	Bajo	Medio	Media	Media
Audio bajo demanda.	Bajo	Bajo	Alta	Baja
Video bajo demanda.	Alto	Bajo	Alta	Baja
Telefonía.	Bajo	Alto	Alta	Baja
Videoconferencias.	Alto	Alto	Alta	Baja

Figura 5-27. Nivel de los requerimientos de calidad del servicio de la aplicación.

Las aplicaciones difieren en cuanto a las necesidades de ancho de banda; las aplicaciones de correo electrónico (*e-mail*), audio en todas las formas e inicio de sesión remoto no necesitan mucho, pero los servicios de compartición de archivos y el video en todas las formas necesitan bastante ancho de banda.

Lo más interesante son los requerimientos de retardo. Las aplicaciones para transferir archivos, incluyendo *e-mail* y video, no son sensibles al retardo. Si todos los paquetes se retardan lo mismo por unos cuantos segundos, no hay problema. Las aplicaciones interactivas, como la navegación web y el inicio de sesión remoto, son más sensibles al retardo. Las aplicaciones en tiempo real, como la telefonía y las videoconferencias, tienen requerimientos estrictos en cuanto al retardo. Si todas las palabras en una llamada telefónica se retardan demasiado, a los usuarios les parecerá inaceptable la conversación. Por otro lado, para reproducir archivos de audio o video de un servidor no se requiere un retardo bajo.

A la variación (es decir, desviación estándar) en el retardo o los tiempos de llegada de los paquetes se le conoce como **variación del retardo** (*jitter*). Las primeras tres aplicaciones en la figura 5-27 no son sensibles a que los paquetes lleguen con intervalos de tiempo irregulares entre ellos. El inicio de sesión remoto es algo sensible a esto, ya que las actualizaciones en la pantalla aparecerán en pequeñas ráfagas si la conexión sufre demasiada variación del retardo. El video y en especial el audio son en extremo sensibles a la variación del retardo. Si un usuario está viendo un video a través de la red y las tramas se retardan exactamente 2 000 segundos, no hay problema. Pero si el tiempo de transmisión varía al azar entre 1 y 2 segundos, el resultado será terrible a menos que la aplicación oculte la variación del retardo. Para el audio, incluso una variación de unos cuantos milisegundos se puede escuchar con claridad.

Las primeras cuatro aplicaciones tienen requerimientos más exigentes sobre la pérdida que el audio y el video, ya que todos los bits se deben entregar correctamente. Por lo general, para lograr este objetivo se retransmiten los paquetes que se pierden en la red mediante la capa de transporte. Esto es trabajo desperdiciado; sería mejor si la red rechazara los paquetes que pudiera llegar a perder en primer lugar. Las aplicaciones de audio y video pueden tolerar unos cuantos paquetes perdidos sin necesidad de retransmitirlos, ya que las personas no detectan las pausas cortas o los saltos ocasionales en las tramas.

Para admitir varias aplicaciones, las redes pueden soportar distintas categorías de QoS. Un ejemplo influyente proviene de las redes ATM, que alguna vez formaron parte de una gran visión para el uso de redes, pero desde entonces se han convertido en una tecnología de nicho. Éstas tienen soporte para:

1. Tasa de bits constantes (por ejemplo, la telefonía).
2. Tasa de bits variable en tiempo real (por ejemplo, videoconferencias con compresión).
3. Tasa de bits variable no en tiempo real (por ejemplo, ver una película bajo demanda).
4. Tasa de bits disponible (por ejemplo, transferencia de archivos).

Estas categorías también son útiles para otros fines y otras redes. La tasa de bits constante es un intento por simular un cable, al proporcionar un ancho de banda y un retardo uniformes. La tasa de bits variable ocurre cuando el video está comprimido y algunas tramas tienen más compresión que otras. Para enviar una trama con muchos detalles en ella, tal vez sea necesario enviar muchos bits, mientras que una toma de una pared blanca se puede comprimir extremadamente bien. Las películas bajo demanda en realidad no son en tiempo real, ya que unos cuantos segundos de video se pueden colocar fácilmente en el búfer del receptor antes de que empiece la reproducción, por lo que la variación del retardo en la red sólo hace que varíe la cantidad de video almacenado que no se ha reproducido todavía. La tasa de bits disponible es para aplicaciones como el correo electrónico, que no son sensibles al retardo o a la variación del retardo, y que toman el ancho de banda que se les asigne.

5.4.2 Modelado de tráfico

Antes de que la red pueda hacer garantías de QoS, debe saber qué tráfico está garantizando. En la red telefónica esta caracterización es simple. Por ejemplo, una llamada de voz (en formato descomprimido) necesita 64 kbps y consiste en una muestra de 8 bits cada 125 μseg. Sin embargo, el tráfico en las redes de datos es **en ráfagas**. Por lo general llega a tasas de transmisión no uniformes a medida que varía el tráfico (es decir, videoconferencias con compresión), los usuarios interactúan con las aplicaciones (por ejemplo, navegar en una nueva página web) y las computadoras cambian de una tarea a otra. Las ráfagas de tráfico son más difíciles de manejar que el tráfico a una tasa constante, ya que pueden llenar búferes y hacer que se pierdan paquetes.

El **modelado de tráfico** (*traffic shaping*) es una técnica para regular la tasa promedio y las ráfagas de un flujo de datos que entra a la red. El objetivo es permitir que las aplicaciones transmitan una amplia variedad de tráfico que se adapte a sus necesidades (incluyendo algunas ráfagas) y tener al mismo tiempo una manera simple y útil de describir los posibles patrones de tráfico a la red. Al establecer un flujo, el usuario y la red (es decir, el cliente y el proveedor) se ponen de acuerdo sobre cierto patrón de tráfico (es decir, la forma) para ese flujo. En efecto, el cliente dice al proveedor: “Mi patrón de transmisión se verá así; ¿puedes manejarlo?”

Algunas veces este acuerdo se denomina **SLA** (**Acuerdo de Nivel de Servicio**, del inglés *Service Level Agreement*), en especial cuando se realiza sobre flujos agregados y periodos extensos, como todo el tráfico para un cliente específico. Mientras que el cliente cumpla con su parte del contrato y sólo envíe los paquetes acordados, el proveedor promete entregarlos de manera oportuna.

El modelado de tráfico reduce la congestión y, por ende, ayuda a la red a cumplir con su promesa. Sin embargo, para que funcione también surge la cuestión de cómo puede saber el proveedor si el cliente está cumpliendo con el acuerdo y qué debe hacer en caso contrario. Los paquetes que excedan el patrón acordado podrían ser descartados por la red, o se podrían marcar con una prioridad más baja. Al monitoreo de un flujo de tráfico se le conoce como **supervisión de tráfico** (*traffic policing*).

El modelado y la supervisión no son tan importantes para las transferencias de igual a igual y otras transferencias similares que consumen todo el ancho de banda disponible, pero son de gran importancia para los datos en tiempo real, como las conexiones de audio y video, que tienen requerimientos exigentes en cuanto a la calidad del servicio.

Cubetas con goteo y con token

Ya vimos una forma de limitar los datos que envía una aplicación: la ventana deslizante, que usa un parámetro para limitar la cantidad de datos en tránsito en un momento dado, lo cual limita la tasa de transmisión en forma indirecta. Ahora veremos una manera más general de caracterizar el tráfico, con los algoritmos de cubeta con goteo y cubeta con token. Las formulaciones son un poco distintas, pero producen un resultado equivalente.

Trate de imaginar una cubeta con un pequeño orificio en el fondo, como se ilustra en la figura 5-28(b). Sin importar la rapidez con que el agua entra a la cubeta, el flujo de salida tiene una tasa constante, *R*, cuando hay agua en la cubeta, y cero cuando la cubeta está vacía. Además, una vez que la cubeta se llena a toda su capacidad *B*, cualquier cantidad adicional de agua que entre se derramará por los costados y se perderá.

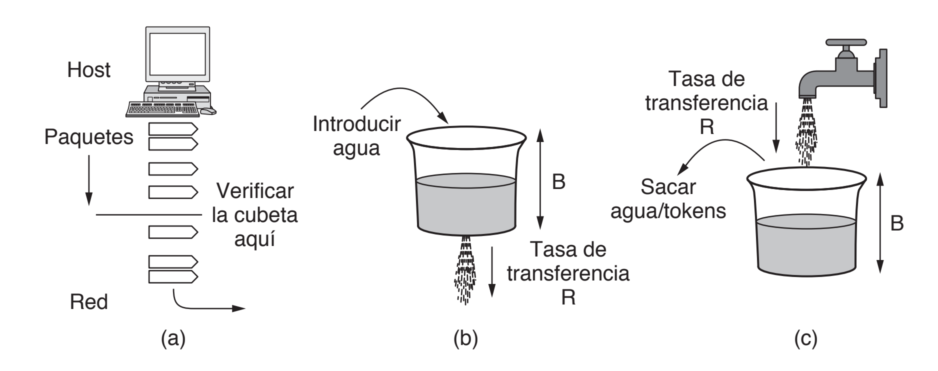


Figura 5-28. (a) Modelado de paquetes. (b) Una cubeta con goteo. (c) Una cubeta con token.

Esta cubeta se puede usar para modelar o supervisar los paquetes que entran a la red, como se muestra en la figura 5-28(a). Conceptualmente, cada host se conecta a la red mediante una interfaz que contiene una cubeta con goteo. Para enviar un paquete a la red, debe ser posible introducir más agua en la cubeta. Si un paquete llega cuando la cubeta está llena, debe ponerse en la cola hasta que se fugue la suficiente cantidad de agua como para poder contenerlo, o se debe descartar. Lo primero podría ocurrir en un host que modela su tráfico para la red como parte del sistema operativo. Lo segundo podría ocurrir en el hardware de una interfaz de red del proveedor, que supervise el tráfico que entra a la red. Turner (1986) propuso esta técnica, conocida como **algoritmo de la cubeta con goteo**.

Una formulación distinta pero equivalente es imaginar la interfaz de red como una cubeta que se está llenando, como se muestra en la figura 5-28(c). El agua fluye por la llave a una tasa de transferencia *R* y la cubeta tiene una capacidad de *B*, como antes. Ahora, para enviar un paquete debemos tener la capacidad

de sacar agua (o tokens, como se le dice normalmente al contenido) de la cubeta (en vez de meter agua en ella). En la cubeta no se puede acumular más de cierta cantidad fija de tokens, *B*, y si la cubeta está vacía, debemos esperar hasta que lleguen más tokens para poder enviar otro paquete. A este algoritmo se le conoce como **algoritmo de la cubeta con tokens**.

Las cubetas con goteo y con tokens limitan la tasa de transferencia a largo plazo de un flujo, pero dejan pasar ráfagas a corto plazo de hasta cierta longitud máxima regulada sin que se alteren ni sufran retardos artificiales. Las ráfagas extensas se controlarán mediante un modelador de tráfico de cubeta con goteo para reducir la congestión en la red. Como ejemplo, imagine que una computadora puede producir datos a una tasa de hasta 1000 Mbps (125 millones de bytes/seg) y que el primer enlace de la red también opera a esta velocidad. El patrón de tráfico que genera el host se muestra en la figura 5-29(a). Este patrón es en ráfagas. La tasa promedio durante un segundo es de 200 Mbps, aun cuando el host envía una ráfaga de 16 000 KB a la máxima velocidad de 1000 Mbps (durante 1/8 de segundo).

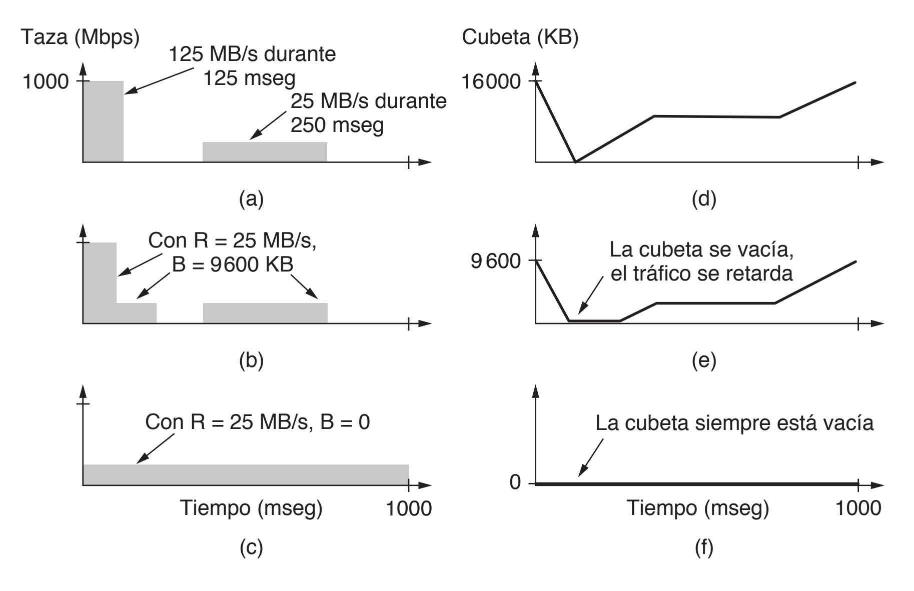


Figura 5-29. (a) Tráfico proveniente de un host. La salida se modela mediante una cubeta de tokens con una tasa de 200 Mbps y con capacidades de (b) 9600 KB y (c) 0 KB. El nivel de la cubeta de tokens para modelar con una tasa de 200 Mbps y capacidades de (d) 16 000 KB, (e) 9 600 KB y (f) 0 KB.

Ahora suponga que los enrutadores pueden aceptar datos a la velocidad máxima sólo durante intervalos cortos, hasta que se llenen sus búferes. El tamaño del búfer es de 9 600 KB, más pequeño que la ráfaga de tráfico. Durante intervalos largos, los enrutadores funcionan mejor con tasas que no se excedan de los 200 Mbps (por decir, ya que éste es todo el ancho de banda que recibe el cliente). La consecuencia es que si el tráfico se envía siguiendo este patrón; una parte se descartará en la red debido a que no cabe en los búferes de los enrutadores.

Para evitar esta pérdida de paquetes, podemos modelar el tráfico en el host mediante una cubeta con tokens. Si usamos una tasa *R,* de 200 Mbps y una capacidad *B* de 9 600 KB, el tráfico se reducirá a una cantidad que la red pueda manejar. La salida de esta cubeta con tokens se muestra en la figura 5-29(b). El host puede enviar con su máxima capacidad a 1000 Mbps durante un intervalo corto, hasta que haya drenado la cubeta. Después tiene que reducir la velocidad a 200 Mbps hasta que se haya enviado la ráfaga. El efecto es dispersar la ráfaga a través del tiempo, ya que era demasiado extensa como para manejarla toda a la vez. El nivel de la cubeta con tokens se muestra en la figura 5-29(e). Empieza llena y se vacía a través

de la ráfaga inicial. Cuando llega a cero, sólo se pueden enviar nuevos paquetes a la tasa a la que se está llenando el búfer; no puede haber más ráfagas sino hasta que la cubeta se haya recuperado. La cubeta se llena cuando no se envía tráfico y permanece sin cambios cuando se envía tráfico a la tasa de llenado.

También podemos modelar el tráfico para que tenga menos ráfagas. La figura 5-29(c) muestra la salida de una cubeta con tokens con $R = 200$ Mbps y una capacidad de 0. Éste es el caso extremo en el que el tráfico se regulariza por completo. No se permiten ráfagas y el tráfico entra a la red a una tasa estable. El nivel de cubeta correspondiente, que se muestra en la figura 5-29(f), siempre está vacío. El tráfico se encola en el host para liberarlo a la red y siempre hay un paquete esperando ser enviado cuando sea posible.

Por último, la figura 5-29(d) muestra el nivel para una cubeta con tokens con $R = 200$ Mbps y una capacidad de $B = 16\,000$ KB. Ésta es la cubeta con tokens más pequeñas por la que el tráfico puede pasar sin sufrir alteraciones. Se podría usar en un enrutador en la red para supervisar el tráfico que envía el host. Si el host envía tráfico que se conforma con base en la cubeta con tokens que acordó con la red, el tráfico pasará por esa misma cubeta de tokens en el enrutador del extremo de la red. Si el host envía a una tasa más rápida o con más ráfagas, la cubeta con tokens se quedará sin agua. Si esto ocurre, un supervisor de tráfico sabrá que el éste no era como se esperaba; entonces descartará los paquetes excesivos o reducirá su prioridad, dependiendo del diseño de la red. En nuestro ejemplo, la cubeta sólo se vacía unos momentos, al final de la ráfaga inicial, y después se recupera lo suficiente para la siguiente ráfaga.

Es fácil implementar las cubetas con goteo y con tokens. Ahora describiremos la operación de una cubeta con token. Aun cuando hemos descrito cómo el agua fluye de manera continua hacia/desde la cubeta, las verdaderas implementaciones deben trabajar con cantidades discretas. Una cubeta con tokens se implementa mediante un contador para el nivel de la cubeta. El contador se incrementa $R/\Delta T$ unidades en cada pulso de reloj de ΔT segundos. En nuestro ejemplo anterior, serían 200 Kbits cada 1 mseg. Cada vez que se envía una unidad de tráfico a la red se decrementa el contador, y se puede enviar tráfico hasta que el contador llegue a cero.

Cuando todos los paquetes son del mismo tamaño, el nivel de la cubeta se puede contar en paquetes (por ejemplo, 200 Mbit equivale a 20 paquetes de 1250 bytes). Sin embargo, es común que se utilicen paquetes de tamaños variables. En este caso, el nivel de la cubeta se cuenta en bytes. Si la cuenta de bytes restantes es demasiado baja como para enviar un paquete extenso, éste debe esperar hasta el siguiente pulso de reloj (o incluso más, si la tasa de llenado es baja).

Es un poco complicado calcular la longitud de la máxima ráfaga (hasta que se vacíe la cubeta). Es más larga que el resultado de dividir 9 600 KB entre 125 MB/seg, ya que mientras se está enviando la ráfaga llegan más tokens. Si designamos la longitud de la ráfaga como S segundos, la tasa máxima de salida como M bytes/seg, la capacidad de la cubeta con tokens como B bytes y la tasa de llegada de tokens como R bytes/seg, podremos ver que una ráfaga de salida contiene un máximo de $B + RS$ bytes. También sabemos que el número de bytes en una ráfaga a máxima velocidad con longitud de S segundos es MS. Por ende, tenemos que

$$B + RS = MS$$

Podemos resolver esta ecuación para obtener $S = B/(M - R)$. Para nuestros parámetros de $B = 9\,600$ KB, $M = 125$ MB/seg y $R = 25$ MB/seg, obtenemos un tiempo de ráfaga aproximado de 94 mseg.

Un problema potencial con el algoritmo de cubeta con tokens es que reduce las ráfagas largas hasta la tasa R de largo plazo. Con frecuencia es deseable reducir la tasa pico, pero sin bajar hasta la tasa de largo plazo (y también sin elevar la tasa de largo plazo para permitir que entre más tráfico en la red). Una forma de obtener un tráfico más uniforme es insertar una segunda cubeta con tokens después de la primera. La tasa de la segunda cubeta deberá ser mucho mayor que la primera. Básicamente, la primera caracteriza el tráfico; corrige su tasa promedio pero permite algunas ráfagas. La segunda reduce la tasa pico a la

que se envían las ráfagas a la red. Por ejemplo, si la tasa de la segunda cubeta con tokens se establece en 500 Mbps y la capacidad en 0, la ráfaga inicial entrará a la red a una tasa pico de 500 Mbps, lo cual es más bajo que la tasa de 1000 Mbps que teníamos antes.

Usar estas cubetas puede ser algo complicado. Cuando se usan con tokens para modelar el tráfico en los hosts, los paquetes se encolan y retardan hasta que las cubetas permitan que se envíen. Cuando se usan cubetas con tokens para supervisar tráfico en los enrutadores en la red, se simula el algoritmo para asegurar que no se envíen más paquetes de los permitidos. Sin embargo, estas herramientas ofrecen maneras de modelar el tráfico de la red en formas más manejables para ayudar a cumplir con los requerimientos de calidad del servicio.

5.4.3 Programación de paquetes

Tener la capacidad de regular la forma del tráfico ofrecido es un buen comienzo. Sin embargo, para ofrecer una garantía de desempeño debemos reservar suficientes recursos a lo largo de la ruta que toman los paquetes a través de la red. Para ello, supongamos que los paquetes de un flujo siguen la misma ruta. Si los dispersamos a través de rutas al azar, es difícil garantizar algo. Como consecuencia, es necesario establecer algo similar a un circuito virtual desde el origen hasta el destino, y todos los paquetes que pertenecen al flujo deben seguir esta ruta.

Los algoritmos que asignan recursos de enrutadores entre los paquetes de un flujo y entre los flujos competidores se llaman **algoritmos de programación de paquetes**. Se pueden reservar potencialmente tres tipos distintos de recursos para diversos flujos:

1. Ancho de banda.
2. Espacio de búfer.
3. Ciclos de CPU.

El primero, ancho de banda, es el más obvio. Si un flujo requiere 1 Mbps y la línea de salida tiene una capacidad de 2 Mbps, no va a ser posible tratar de dirigir tres flujos a través de esa línea. Por lo tanto, reservar ancho de banda significa no sobrecargar ninguna línea de salida.

Un segundo recurso que por lo general escasea es el espacio en búfer. Cuando llega un paquete, se pone en un búfer dentro del enrutador hasta que se pueda transmitir en la línea de salida elegida. El propósito del búfer es absorber pequeñas ráfagas de tráfico a medida que los flujos compiten entre sí. Si no hay búfer disponible, el paquete se tiene que descartar debido a que no hay lugar para colocarlo. Para una buena calidad de servicio, se deberían reservar algunos búferes para un flujo específico de manera que éste no tenga que competir con otros flujos para obtener el espacio del búfer. Siempre que ese flujo necesite un búfer, se le proporcionará uno hasta cierto valor máximo.

Por último, los ciclos de CPU también pueden ser un recurso escaso. Para procesar un paquete se necesita tiempo de CPU del enrutador, por lo que un enrutador sólo puede procesar cierta cantidad de paquetes por segundo. Aunque los enrutadores modernos pueden procesar la mayoría de los paquetes rápidamente, algunos tipos de paquetes requieren más procesamiento de la CPU, como los paquetes ICMP que describiremos en la sección 5.6. Para asegurar el procesamiento oportuno de estos paquetes, es necesario asegurarse que la CPU no esté sobrecargada.

Para asignar ancho de banda y otros recursos del enrutador, los algoritmos de programación de paquetes determinan cuál de los paquetes en el búfer van a enviar en la línea de salida a continuación. Ya describimos el programador más simple al explicar cómo funcionan los enrutadores. Cada enrutador coloca los paquetes en una cola de búfer para cada línea de salida hasta que se puedan enviar, y se envían en el mismo orden en el que llegaron. Este algoritmo se conoce como **FIFO (Primero en Entrar, Primero en**

Salir, del inglés *First-In First-Out*) o su equivalente, **FCFS** (**Primero en Llegar, Primero en Servir**, del inglés *First-Come, First-Serve*).

Por lo general, los enrutadores FIFO descartan los paquetes recién llegados cuando la cola está llena. Como el paquete recién llegado se hubiera colocado al final de la cola, a este comportamiento se le conoce como **descarte trasero (***tail drop***)**. Es intuitivo y tal vez se pregunte qué alternativas existen. De hecho, el algoritmo RED que describimos en la sección 5.3.5 elegía un paquete recién llegado para descartarlo al azar cuando la longitud promedio de la cola aumentaba mucho. Los otros algoritmos de programación que describiremos también crean otras oportunidades para decidir qué paquete descartar cuando los búferes están llenos.

La programación FIFO es simple de implementar, pero no es adecuada para proporcionar una buena calidad del servicio, ya que cuando hay varios flujos, uno de ellos puede afectar fácilmente el desempeño de los demás. Si el primer flujo es agresivo y envía grandes ráfagas de paquetes, se depositarán en la cola. Procesar paquetes con base en su orden de llegada significa que el emisor agresivo puede acaparar la mayor parte de la capacidad de los enrutadores que sus paquetes recorren, privando a los demás flujos y reduciendo su calidad de servicio. Para empeorar las cosas, es probable que los paquetes de los otros flujos que logren pasar se retarden, ya que tuvieron que sentarse en la cola detrás de muchos paquetes provenientes del emisor agresivo.

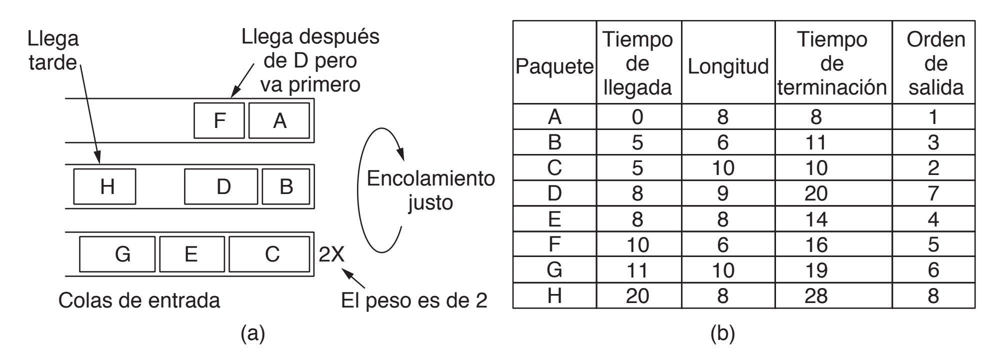
Se han ideado muchos algoritmos de programación de paquetes que proporcionan un aislamiento más sólido entre los flujos y frustran los intentos de interferencia (Bhatti y Crowcroft, 2000). Uno de los primeros fue el algoritmo de **encolamiento justo** ideado por Nagle (1987). La esencia de este algoritmo es que los enrutadores tienen colas separadas, una por cada flujo para una línea de salida dada. Cuando la línea se vuelve inactiva, el enrutador explora de forma circular (*round-robin*) las colas, como se muestra en la figura 5-30. Entonces toma el primer paquete de la siguiente cola. Así, con n hosts que compiten por la línea de salida, cada host tiene la oportunidad de enviar uno de cada n paquetes. Es justo en el sentido de que todos los flujos pueden enviar paquetes a la misma tasa. Enviar más paquetes no mejorará esta tasa.



Figura 5-30. Encolamiento justo por exploración circular (*round-robin*).

Aunque es un comienzo, este algoritmo tiene una falla: proporciona más ancho de banda a los hosts que utilizan paquetes grandes que a los que utilizan paquetes pequeños. Demers y colaboradores (1990) sugirieron una mejora en la que la exploración circular (*round-robin*) se realiza de tal forma que se simule una exploración circular byte por byte, en vez de paquete por paquete. El truco es calcular un tiempo virtual que sea el número de la ronda en la que cada paquete terminará de ser enviado. Cada ronda drena un byte de todas las colas que tienen datos por enviar. Entonces los paquetes se ordenan conforme a su tiempo de terminación y se envían en ese orden.

En la figura 5-31 se muestra este algoritmo con un ejemplo de los tiempos de terminación para los paquetes que llegan en tres flujos. Si un paquete tiene una longitud L, la ronda en la que terminará es simplemente L rondas después del tiempo de inicio, que puede ser el tiempo de terminación del paquete anterior o el tiempo de llegada del paquete, si la cola está vacía cuando llega.



Paquete	Tiempo de llegada	Longitud	Tiempo de terminación	Orden de salida
A	0	8	8	1
B	5	6	11	3
C	5	10	10	2
D	8	9	20	7
E	8	8	14	4
F	10	6	16	5
G	11	10	19	6
H	20	8	28	8

Figura 5-31. (a) Encolamiento justo ponderado. (b) Tiempos de terminación para los paquetes.

Si observamos la tabla de la figura 5-31(b), y analizamos sólo los primeros dos paquetes en las dos colas superiores, nos daremos cuenta de que los paquetes llegan en el orden *A*, *B*, *D* y *F*. El paquete *A* llega en la ronda 0 y tiene 8 bytes de longitud, por lo que su tiempo de terminación es en la ronda 8. De manera similar, el tiempo de terminación para el paquete *B* es 11. El paquete *D* llega mientras se está enviando *B*. Su tiempo de terminación es 9 rondas de bytes después de que empieza cuando *B* termina, o 20. Asimismo, el tiempo de terminación para *F* es 16. Como no llegan paquetes nuevos, el orden de envío relativo es *A*, *B*, *F*, *D*, aun cuando *F* llegó después de *D*. Es posible que llegue otro pequeño paquete en el flujo superior y que obtenga un tiempo de terminación antes que *D*. Sólo se adelantará a *D* si no ha empezado la transmisión de ese paquete. El encolamiento justo no sustituye los paquetes que ya se están transmitiendo. Como los paquetes se envían en su totalidad, el encolamiento justo sólo es una aproximación del esquema ideal de byte por byte. Pero es una muy buena aproximación que en todo momento se mantiene con una diferencia de un paquete con respecto a la transmisión mediante el esquema ideal.

Una desventaja de este algoritmo en la práctica es que da la misma prioridad a todos los hosts. En muchas situaciones es conveniente dar a los servidores de video más ancho de banda que a los servidores de archivos, por dar un ejemplo. Para hacer esto sólo es necesario dar al servidor de video dos o más bytes por ronda. Este algoritmo modificado se llama **WFQ** (**Encolamiento Justo Ponderado**, del inglés *Weighted Fair Queueing*). Si dejamos que el número de bytes por ronda sea el peso de un flujo, *W*, podemos obtener la fórmula para calcular el tiempo de terminación:

$$F_i = \text{máx}(A_i, F_{i-1}) + L_i/W$$

en donde A_i es el tiempo de llegada, F_i es el tiempo de terminación y L_i es la longitud del paquete *i*. La cola inferior de la figura 5-31(a) tiene un peso de 2, por lo que sus paquetes se envían con más rapidez como podemos ver en los tiempos de terminación que se muestran en la figura 5-31(b).

Otra consideración práctica es la complejidad de la implementación. WFQ requiere que los paquetes se inserten en una cola ordenada, con base en su tiempo de terminación. Con *N* flujos, ésta es a lo más una operación $O(\log N)$ por paquete, lo cual es difícil de lograr para muchos flujos en los enrutadores de alta velocidad. Shreedhar y Varghese (1995) describen una aproximación conocida como *déficit de exploración circular* (*deficit round robin*) que se puede implementar en forma muy eficiente, con sólo $O(1)$ operaciones por paquete. WFQ se utiliza mucho debido a esta aproximación.

También existen otros tipos de algoritmos de programación. La programación por prioridad es un ejemplo simple, en donde cada paquete se marca con una prioridad. Los paquetes de prioridad alta siempre se envían antes que los de prioridad baja que estén en el búfer. Si tienen la misma prioridad, los paquetes se envían en orden FIFO. Sin embargo, la programación por prioridad tiene la desventaja de que una ráfaga de paquetes de alta prioridad puede privar a los paquetes de baja prioridad, que tal vez

tengan que esperar de manera indefinida. WFQ ofrece por lo general una mejor alternativa. Al asignar a la cola de alta prioridad un peso grande (por decir, 3), por lo general los paquetes de alta prioridad pasarán a través de una línea corta (puesto que relativamente debe haber pocos paquetes de alta prioridad) y se seguirá enviando cierta fracción de paquetes de baja prioridad, aun cuando haya tráfico de alta prioridad. Un sistema de baja y alta prioridad es en esencia un sistema WFQ de dos colas, en donde la prioridad alta tiene un peso infinito.

Como ejemplo final de un programador, los paquetes podrían incluir etiquetas de tiempo para enviarse en ese orden. Clark y colaboradores (1992) describen un diseño en el que la etiqueta de tiempo registra qué tan atrasado o adelantado está el paquete según la programación, a medida que se envía a través de una secuencia de enrutadores por la ruta designada. Los paquetes que se hayan puesto en cola detrás de otros paquetes en un enrutador tenderán a estar atrasados, y los paquetes que se hayan atendido primero tenderán a estar adelantados. El efecto benéfico de enviar paquetes con base en el orden de sus etiquetas de tiempo es que los paquetes lentos se agilizan, al tiempo que se reduce la velocidad de los paquetes rápidos. El resultado es que la red entrega todos los paquetes con un retardo más consistente.

5.4.4 Control de admisión

Ya hemos visto todos los elementos necesarios para la calidad del servicio (QoS); es tiempo de reunirlos para realmente proporcionarla. Las garantías de QoS se establecen por medio del proceso de control de admisión. Primero vimos cómo se utilizaba el control de admisión para controlar la congestión, lo cual es una garantía de desempeño, aunque algo débil. Las garantías que consideraremos ahora son más sólidas, pero el modelo es el mismo. El usuario ofrece un flujo junto con un requerimiento de QoS para la red. Después la red decide si acepta o rechaza el flujo, con base en su capacidad y a los compromisos que tiene con otros flujos. Si acepta, reserva la capacidad por adelantado en los enrutadores para garantizar la QoS cuando se envíe el tráfico por el nuevo flujo.

Es necesario hacer las reservaciones en todos los enrutadores a lo largo de la ruta que toman los paquetes a través de la red. Cualquier enrutador que esté en la ruta sin reservaciones podría congestionarse, y un solo enrutador congestionado puede quebrantar la garantía de QoS. Muchos algoritmos de enrutamiento encuentran la mejor ruta entre cada origen y cada destino, y envían todo el tráfico a través de la mejor ruta. Esto puede ocasionar que se rechacen algunos flujos si es que no hay suficiente capacidad disponible a lo largo de la mejor ruta. De todas formas, tal vez sea posible adoptar las garantías de QoS para los nuevos flujos si se elije una ruta distinta para el flujo con la capacidad excesiva. Esto se llama **enrutamiento con QoS**. Chen y Nahrstedt (1998) indican las generalidades sobre estas técnicas. También es posible dividir el tráfico para cada destino a través de varias rutas, de modo que sea más fácil encontrar la capacidad adicional. Un método simple es que los enrutadores seleccionen rutas de igual costo y dividan el tráfico en forma equitativa, o en proporción a la capacidad de los enlaces de salida. Sin embargo, también existen algoritmos más sofisticados (Nelakuditi y Zhang, 2002).

Dada una ruta, la decisión de aceptar o rechazar un flujo no es una simple cuestión de comparar los recursos (ancho de banda, búfer, ciclos) solicitados por el flujo con la capacidad excesiva del enrutador en esas tres dimensiones. Es un poco más complicado que eso. Para empezar, aunque tal vez ciertas aplicaciones conozcan sus requerimientos de ancho de banda, pocas conocen sobre los requerimientos de búfer o ciclos de CPU, por lo que como mínimo se requiere una manera distinta de describir los flujos y traducir esta descripción para los recursos del enrutador. En breve trataremos esta cuestión.

Ahora bien, algunas aplicaciones son más tolerantes que otras en cuanto a no cumplir con uno que otro plazo límite. Las aplicaciones deben elegir de entre el tipo de garantías que puede hacer la red, ya sean garantías rígidas o un comportamiento que sea suficiente la mayor parte del tiempo. Si todo lo demás fuera igual, a todos les gustaría trabajar con garantías rígidas, pero la dificultad es que son costosas debido

a que restringen el comportamiento en el peor de los casos. Con frecuencia las garantías para la mayoría de los paquetes son suficientes para las aplicaciones, y se pueden soportar más flujos con esta clase de garantía para una capacidad fija.

Por último, tal vez algunas aplicaciones estén dispuestas a regatear en cuanto a los parámetros de los flujos, pero otras no. Por ejemplo, un cinéfilo que por lo general vea películas a 30 tramas/seg podría estar dispuesto a cambiar a 25 tramas/seg si no hay suficiente ancho de banda como para soportar 30 tramas/seg. De manera similar, tal vez se puedan ajustar el número de píxeles por trama, el ancho de banda de audio y otras propiedades.

Como puede haber muchas partes involucradas en la negociación del flujo (el emisor, el receptor y todos los enrutadores a lo largo de la ruta entre ellos), debemos describir los flujos con precisión en términos de los parámetros específicos que se pueden negociar. Un conjunto de tales parámetros se denomina **especificación de flujo**. Por lo general, el emisor (por ejemplo, el servidor de video) produce una especificación de flujo que propone los parámetros que le gustaría usar. A medida que se propaga la especificación a lo largo de la ruta, cada enrutador la examina y modifica los parámetros según sus necesidades. Las modificaciones sólo pueden reducir el flujo, no incrementarlo (por ejemplo, una tasa de datos más baja, no una más alta). Cuando la especificación llega al otro extremo, se pueden establecer los parámetros.

Como ejemplo de lo que puede haber en una especificación de flujo, considere el de la figura 5-32, que se basa en los RFC 2210 y 2211 para los Servicios integrados, un diseño de QoS que cubriremos en la siguiente sección. Tiene cinco parámetros. Los primeros dos, la *tasa de la cubeta con tokens* y el *tamaño de la cubeta con tokens*, utilizan una cubeta con tokens para proporcionar la tasa máxima sostenida a la que puede transmitir el emisor, promediada con respecto a un intervalo largo y la ráfaga más grande que puede enviar durante un intervalo corto.

Parámetro	Unidad
Tasa de la cubeta con tokens.	Bytes/seg
Tamaño de la cubeta con tokens.	Bytes
Tasa pico de datos.	Bytes/seg
Tamaño mínimo de paquete.	Bytes
Tamaño máximo de paquete.	Bytes

Figura 5-32. Ejemplo de una especificación de flujo.

El tercer parámetro, la *tasa pico de datos*, es la tasa máxima de transmisión tolerada, incluso durante intervalos breves. El emisor nunca debe sobrepasar esta tasa, ni siquiera en ráfagas cortas.

Los últimos dos parámetros especifican los tamaños mínimo y máximo de paquetes, incluyendo los encabezados de la capa de transporte y de red (por ejemplo, TCP e IP). El tamaño mínimo es importante porque procesar cada paquete toma un tiempo fijo, aunque sea corto. Un enrutador podrá estar preparado para manejar 10 000 paquetes/seg de 1 KB cada uno, pero no para manejar 100 000 paquetes/seg de 50 bytes cada uno, aunque esto represente una tasa de datos menor. El tamaño máximo de paquete es importante debido a las limitaciones internas de la red que no deben sobrepasarse. Por ejemplo, si parte de la ruta pasa a través de una Ethernet, el tamaño máximo del paquete se restringirá a no más de 1500 bytes, sin importar lo que el resto de la red pueda manejar.

Una pregunta interesante es cómo convierte un enrutador una especificación de flujo en un conjunto de reservaciones de recursos específicos. A primera instancia, podría parecer que si un enrutador tiene un enlace que opera a, por decir, 1 Gbps y el paquete promedio es de 1000 bits, puede procesar 1 millón de

paquetes/seg. Pero esta observación no es verdadera, ya que siempre habrá periodos inactivos en el enlace debido a las fluctuaciones estadísticas en la carga. Si el enlace necesita toda la capacidad disponible para realizar su trabajo, al estar inactivo incluso por unos cuantos bits se generará una acumulación de la que nunca se podrá deshacer.

Incluso con una carga ligeramente por debajo de la capacidad teórica, se pueden generar colas y pueden ocurrir retardos. Considere una situación en la que los paquetes llegan de manera aleatoria con una tasa promedio de llegada de λ paquetes/seg. Los paquetes tienen longitudes aleatorias y se pueden enviar por el enlace con una tasa promedio de servicio de μ paquetes/seg. Bajo el supuesto de que las distribuciones de arribo y de servicio son distribuciones Poisson (lo cual se conoce como sistema de encolamiento M/M/1, en donde "M" es de Markov, es decir, Poisson), es posible probar mediante la teoría de colas que el retardo promedio experimentado por un paquete T es

$$T = \frac{1}{\mu} \times \frac{1}{1 - \lambda/\mu} = \frac{1}{\mu} \times \frac{1}{1 - \rho}$$

donde $\rho = \lambda/\mu$ es el uso de CPU. El primer factor, $1/\mu$, es lo que sería el tiempo de servicio si no hubiera competencia. El segundo factor es la reducción de velocidad debido a la competencia con otros flujos. Por ejemplo, si $\lambda = 950\,000$ paquetes/seg y $\mu = 1\,000\,000$ paquetes/seg, entonces $\rho = 0.95$ y el retardo promedio experimentado por cada paquete será de 20 μseg en lugar de 1 μseg. Este tiempo cuenta tanto para el tiempo de encolamiento como para el de servicio, como puede verse cuando la carga es muy baja ($\lambda/\mu = 0$). Si hay, por ejemplo, 30 enrutadores a lo largo de la ruta del flujo, tan sólo el retardo de encolamiento será de alrededor de 600 μseg.

Parekh y Gallagher (1993, 1994) idearon un método para relacionar las especificaciones de flujo con los recursos de enrutador que correspondan con las garantías de desempeño de ancho de banda y retardo. Este método se basa en modelar las fuentes de tráfico mediante cubetas de tokens (R, B) y WFQ en los enrutadores. Cada flujo recibe un peso W de WFQ lo suficientemente grande como para drenar su cubeta de tokens a la tasa R, como se muestra en la figura 5-33. Por ejemplo, si el flujo tiene una tasa de 1 Mbps y tanto el enrutador como el enlace de salida tienen una capacidad de 1 Gbps, el peso para el flujo debe ser mayor que 1/1000 parte del total de los pesos para todos los flujos en ese enrutador para el enlace de salida. Esto garantiza al flujo un ancho de banda mínimo. Si no puede proporcionar una tasa suficientemente grande, no se puede admitir el flujo.

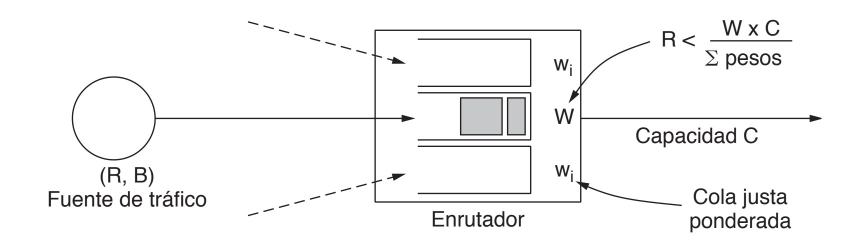


Figura 5-33. Garantías de ancho de banda y retardo mediante cubetas con tokens y WFQ.

El retardo de encolamiento más grande que verá el flujo es en función del tamaño de ráfaga de la cubeta con tokens. Considere los dos casos extremos. Si el tráfico es uniforme y sin ráfagas, los paquetes se drenarán del enrutador con la misma rapidez con que llegan. No habrá retardo de encolamiento (ignorando los efectos de la paquetización). Por otro lado, si el tráfico se acumula en ráfagas, entonces puede llegar una ráfaga de tamaño máximo B al enrutador, toda a la vez. En este caso, el retardo de encolamiento

máximo D será el tiempo requerido para drenar esta ráfaga con base en el ancho de banda garantizado, o B/R (de nuevo, ignorando los efectos de la paquetización). Si este retardo es demasiado grande, el flujo debe solicitar más ancho de banda a la red.

Estas garantías son rígidas. Las cubetas con tokens limitan las ráfagas de la fuente y el encolamiento justo aísla el ancho de banda que se proporciona a los distintos flujos. Esto significa que el flujo cumplirá con sus garantías de ancho de banda y retardo sin importar cómo se comporten los demás flujos competidores en el enrutador. Esos otros flujos no podrán quebrantar la garantía, ni siquiera al acumular tráfico y enviarlo todo a la vez.

Además, el resultado es válido para una ruta a través de varios enrutadores en cualquier topología de red. Cada flujo obtiene un ancho de banda mínimo, ya que ese ancho de banda está garantizado en cada enrutador. La razón por la que cada flujo obtiene un retardo máximo es más sutil. En el peor caso en que una ráfaga de tráfico llegue al primer enrutador y compita con el tráfico de otros flujos, se retardará hasta un tiempo máximo de D. Sin embargo, este retardo también regulará la ráfaga. A su vez, esto significa que la ráfaga no incurrirá en más retardos de encolamiento en los siguientes enrutadores. A lo más, el retardo de encolamiento general será de D.

5.4.5 Servicios integrados

Entre 1995 y 1997, la IETF se esforzó mucho en diseñar una arquitectura para la multimedia de flujos continuos. Este trabajo generó cerca de dos docenas de RFC, empezando con los RFC 2205–2212. El nombre genérico para este trabajo es **servicios integrados**. Se diseñó tanto para aplicaciones de unidifusión como para multidifusión. Un ejemplo de la primera es un solo usuario recibiendo en flujo continuo un clip de video de un sitio de noticias. Un ejemplo de la segunda es una colección de estaciones de televisión digital difundiendo sus programas como flujos de paquetes IP a muchos receptores en distintas ubicaciones. A continuación nos concentraremos en la multidifusión, puesto que la unidifusión es un caso especial de la multidifusión.

En muchas aplicaciones de multidifusión, los grupos pueden cambiar su membresía en forma dinámica; por ejemplo, conforme las personas entran a una videoconferencia, se aburren y cambian a una telenovela o al canal del juego de criquet. Bajo estas condiciones, el método de hacer que los emisores reserven ancho de banda por adelantado no funciona bien, debido a que requeriría que cada emisor rastreara todas las entradas y salidas de su audiencia. En un sistema diseñado para transmitir televisión con millones de suscriptores, ni siquiera funcionaría.

RSVP: el protocolo de reservación de recursos

La parte principal de la arquitectura de servicios integrados visible para los usuarios de la red es **RSVP**. Se describe en los RFC 2205-2210. Este protocolo se utiliza para hacer las reservaciones; se usan otros protocolos para enviar los datos. RSVP brinda la posibilidad de que varios emisores transmitan a múltiples grupos de receptores, permite que receptores individuales cambien de canal libremente, optimiza el uso de ancho de banda y al mismo tiempo elimina la congestión.

En su forma más simple, el protocolo usa enrutamiento de multidifusión con árboles de expansión, como vimos antes. A cada grupo se le asigna una dirección. Para enviar a un grupo, un emisor pone la dirección asignada en sus paquetes. El algoritmo estándar de enrutamiento multidifusión construye entonces un árbol de expansión que cubre a todos los miembros del grupo. El algoritmo de enrutamiento no es parte de RSVP. La única diferencia con la multidifusión normal es un poco de información adicional que se transmite por multidifusión al grupo en forma periódica, para indicarle a los enrutadores a lo largo del árbol que mantengan ciertas estructuras de datos en sus memorias.

Como ejemplo, considere la red de la figura 5-34(a). Los hosts 1 y 2 son emisores multidifusión, y los hosts 3, 4 y 5 son receptores multidifusión. En este ejemplo, los emisores y los receptores están separados pero, en general, los dos grupos se pueden traslapar. Los árboles de multidifusión de los hosts 1 y 2 se muestran en las figuras 5-34(b) y 5-34(c), respectivamente.

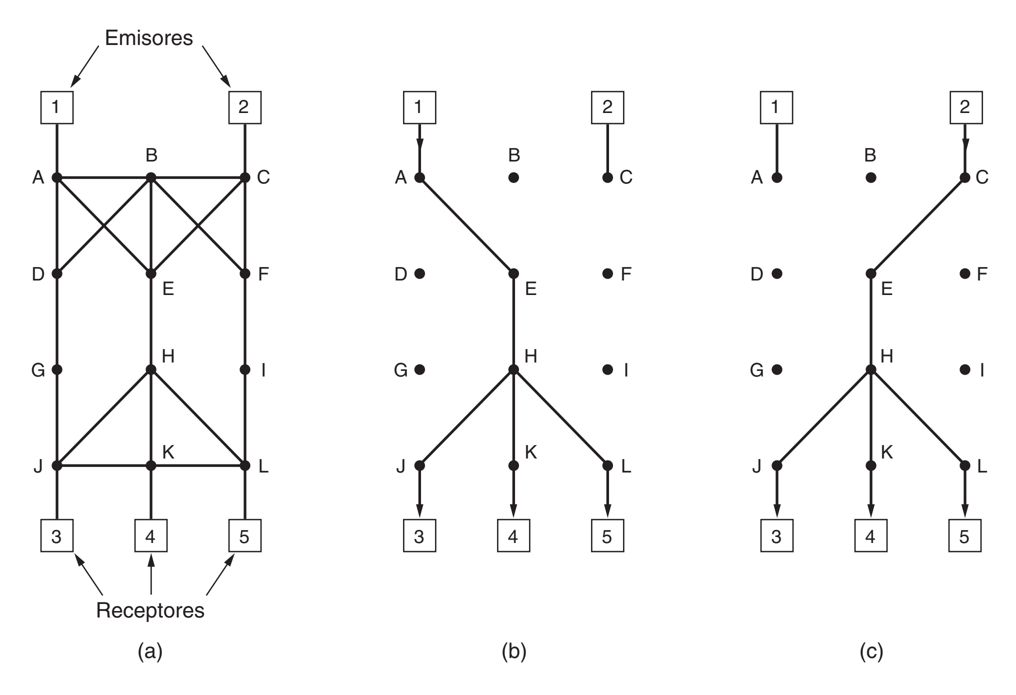


Figura 5-34. (a) Una red. (b) El árbol de expansión de multidifusión para el host 1. (c) El árbol de expansión de multidifusión para el host 2.

Para obtener una mejor recepción y eliminar la congestión, cualquiera de los receptores en un grupo puede enviar un mensaje de reservación al emisor a través del árbol. El mensaje se propaga mediante el algoritmo de reenvío por ruta invertida que vimos antes. En cada salto, el enrutador nota la reservación y aparta el ancho de banda necesario. En la sección anterior vimos cómo se puede usar un programador de encolamiento justo ponderado para hacer esa reservación. Si no hay suficiente ancho de banda disponible, informa sobre una falla. Para cuando el mensaje regresa a la fuente, se ha reservado ancho de banda por toda la trayectoria desde el emisor hasta el receptor que hace la solicitud de reservación a lo largo del árbol de expansión.

En la figura 5-35(a) se muestra un ejemplo de tales reservaciones. Aquí el host 3 ha solicitado un canal al host 1. Una vez establecido el canal, los paquetes pueden fluir de 1 a 3 sin congestiones. Ahora considere lo que ocurre si el host 3 reserva a continuación un canal hacia otro emisor, el host 2, para que el usuario pueda ver dos programas de televisión a la vez. Se reserva una segunda ruta, como se muestra en la figura 5-35(b). Cabe mencionar que se requieren dos canales individuales del host 3 al enrutador *E* porque se están transmitiendo dos flujos independientes.

Por último, en la figura 5-35(c) el host 5 decide observar el programa transmitido por el host 1 y también hace una reservación. Primero se reserva ancho de banda dedicado hasta el enrutador *H*. Sin embargo, éste ve que ya tiene una alimentación del host 1, por lo que si ya se ha reservado el ancho de banda necesario, no necesita reservar más. Observe que los hosts 3 y 5 podrían haber solicitado diferentes cantidades de ancho de banda (por ejemplo, el host 3 tiene una televisión pequeña y sólo quiere la infor-

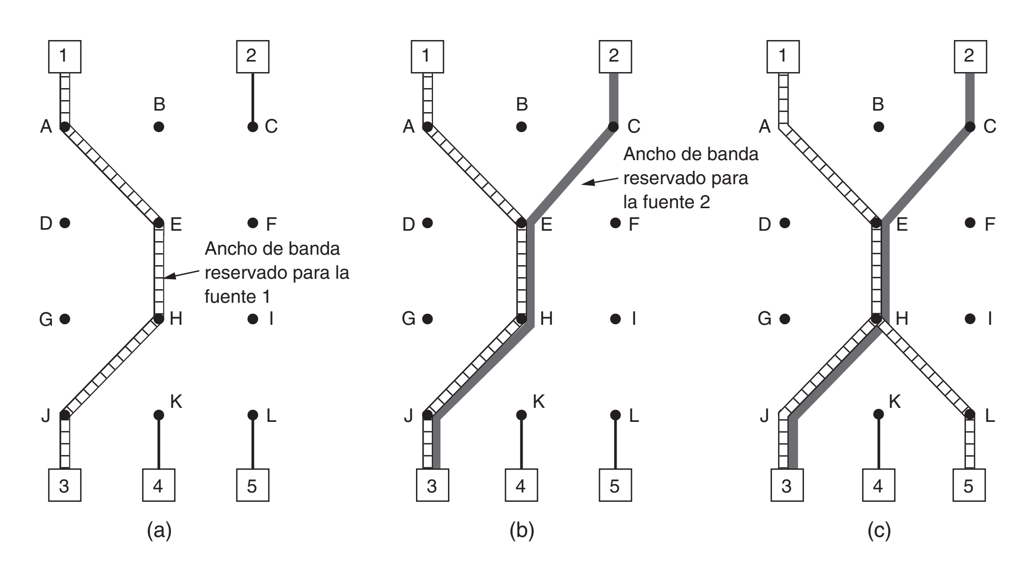


Figura 5-35. (a) El host 3 solicita un canal al host 1. (b) Después el host 3 solicita un segundo canal, al host 2. (c) El host 5 solicita un canal al host 1.

mación de baja resolución), así que la capacidad reservada debe ser lo bastante grande como para satisfacer hasta el receptor más voraz.

Al hacer una reservación, un receptor puede especificar (de manera opcional) una o más fuentes de las que desea recibir información. También puede especificar si estas selecciones quedarán fijas durante toda la reservación, o si el receptor quiere mantener abierta la opción de cambiar después las fuentes. Los enrutadores usan esta información para optimizar la planeación del ancho de banda. En particular, sólo se establecen dos receptores para compartir una ruta si ambos están de acuerdo con no cambiar las fuentes posteriormente.

La razón de esta estrategia en el caso totalmente dinámico es que el ancho de banda reservado está desacoplado de la selección de la fuente. Una vez que un receptor ha reservado ancho de banda, puede cambiar a otra fuente y conservar esa parte de la ruta existente que sea válida para la nueva fuente. Si el host 2 está transmitiendo varios flujos de video en tiempo real (como un difusor de TV con varios canales), el host 3 puede cambiar entre ellos a voluntad sin modificar su reservación: a los enrutadores no les importa el programa que está viendo el receptor.

5.4.6 Servicios diferenciados

Los algoritmos basados en flujo tienen el potencial de ofrecer buena calidad de servicio a uno o más flujos, debido a que reservan los recursos necesarios a lo largo de la ruta. Sin embargo, también tienen una desventaja. Requieren una configuración avanzada para establecer cada flujo, algo que no se escala bien cuando hay miles o millones de flujos. Además, mantienen el estado por flujo interno en los enrutadores, lo cual los hace vulnerables a las fallas de éstos. Por último, los cambios requeridos al código de enrutador son considerables e involucran intercambios complejos de enrutador a enrutador para establecer los flujos. Como consecuencia, aunque el trabajo continúa para mejorar los servicios integrados, existen pocas implementaciones de RSVP o algo parecido.

Por estas razones, la IETF también ha diseñado un método más simple para la calidad del servicio, uno que se pueda implementar ampliamente de manera local en cada enrutador sin una configuración avanzada y sin que toda la ruta esté involucrada. Este método se conoce como calidad de servicio **basada**

en clase (en contraste a la basada en flujo). La IETF ha estandarizado una arquitectura para él, conocida como **servicios diferenciados**, que se describe en los RFC 2474, 2475 y varios más. A continuación lo describiremos.

Un conjunto de enrutadores que forman un dominio administrativo (por ejemplo, un ISP o una compañía telefónica) puede ofrecer los servicios diferenciados. La administración define un conjunto de clases de servicios con sus correspondientes reglas de reenvío. Si un cliente se suscribe a los servicios diferenciados, los paquetes del cliente que entran en el dominio se marcan con la clase a la que pertenecen. Esta información se transporta en el campo Servicios diferenciados (*Differentiated services*) de los paquetes IPv4 e IPv6 (descritos en la sección 5.6). Las clases se definen como **comportamientos por salto**, ya que corresponden al trato que recibirá el paquete en cada enrutador y no a una garantía a través de la red. Se proporciona un mejor servicio a los paquetes con ciertos comportamientos por salto (por ejemplo, servicio premium) que a otros paquetes (por ejemplo, servicio regular). Al tráfico dentro de una clase se le podría requerir que se apegue a algún modelo específico, como a una cubeta con goteo y con cierta tasa especificada de drenado. Un operador con intuición para los negocios podría cobrar una cantidad adicional por cada paquete premium transportado, o podría permitir hasta *N* paquetes premium al mes por una mensualidad adicional fija. Cabe mencionar que este esquema no requiere una configuración avanzada o una reserva de recursos, ni tampoco una negociación extremo a extremo que consuma tiempo para cada flujo, como sucede con los servicios integrados. Esto hace que los servicios diferenciados sean relativamente sencillos de implementar.

El servicio basado en clase también ocurre en otras industrias. Por ejemplo, las compañías de envío de paquetes con frecuencia ofrecen servicio de tres días, de dos días, y servicio de un día para otro. Las aerolíneas ofrecen servicio de primera clase, de clase de negocios y de tercera clase. Los trenes que recorren largas distancias con frecuencia tienen múltiples clases de servicios. Incluso el metro de París tiene dos clases distintas de servicios. Para los paquetes, las clases pueden diferir en términos de retardo, variación del retardo y probabilidad de que se descarten en caso de congestión, entre otras posibilidades (pero es probable que sin tramas Ethernet más amplias).

Para hacer que la diferencia entre la calidad de servicio basada en flujo y la basada en clase sea más clara, considere un ejemplo: la telefonía de Internet. Con un esquema basado en flujo, cada llamada telefónica obtiene sus propios recursos y garantías. Con un esquema basado en clase, todas las llamadas telefónicas obtienen los recursos reservados para la clase telefonía. Los paquetes de clase de navegación web o de otras clases no pueden tomar estos recursos, pero ninguna llamada telefónica puede obtener ningún recurso privado reservado sólo para ella.

Reenvío expedito

Cada operador debe elegir las clases de servicios, pero ya que los paquetes con frecuencia se reenvían entre redes operadas por diferentes operadores, la IETF ha definido algunas clases de servicios independientes de la red. La clase más simple es el **reenvío expedito**, por lo que iniciaremos con ella. Se describe en el RFC 3246.

La idea detrás del reenvío expedito es muy simple. Hay dos clases de servicios disponibles: regular y expedito. Se espera que la mayor parte del tráfico sea regular, pero una pequeña fracción de los paquetes son expeditos. Éstos deben tener la capacidad de transitar por la red como si no hubiera otros paquetes. De esta forma ellos tendrán un servicio con pocas pérdidas, bajo retardo y poca variación del retardo; justo lo que VoIP necesita. En la figura 5-36 se muestra una representación simbólica de este sistema de "dos tubos". Observe que todavía hay sólo una línea física. Las dos tuberías lógicas que se muestran en la figura representan una forma de reservar ancho de banda para las distintas clases de servicio, no una segunda línea física.

A continuación veremos una forma de implementar esta estrategia. Los paquetes se clasifican como expeditos o regulares y se marcan según corresponda. Este paso se puede realizar en el host emisor o en el

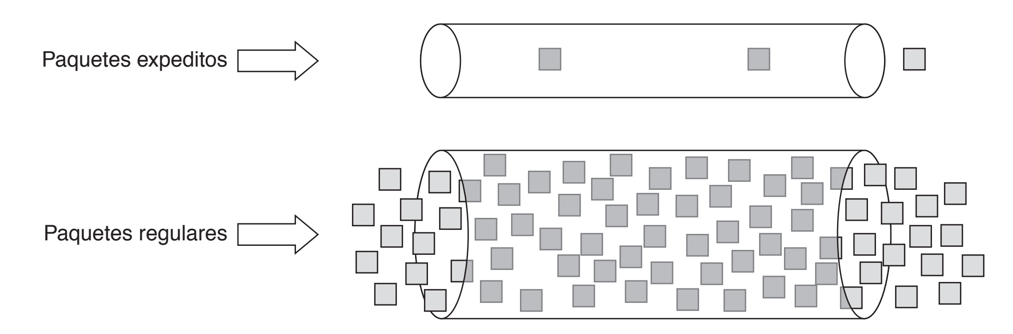


Figura 5-36. Los paquetes expeditos experimentan una red sin tráfico.

enrutador de ingreso (el primero). La ventaja de realizar la clasificación en el host emisor es que hay más información disponible acerca de los paquetes que pertenecen a cada flujo. Esta tarea se puede llevar a cabo mediante software de red o incluso a través del sistema operativo, para no tener que cambiar las aplicaciones existentes. Por ejemplo, cada vez es más común que los hosts marquen los paquetes VoIP para un servicio expedito. Si los paquetes pasan a través de una red corporativa o un ISP que soporte el servicio expedito, recibirán un trato especial. Si la red no soporta el servicio expedito, no habrá ningún problema.

Desde luego que si el host se encarga de marcar los paquetes, es probable que el enrutador de ingreso supervise el tráfico para asegurarse que los clientes no envíen más tráfico expedito del que pagaron. Dentro de la red, los enrutadores pueden tener dos colas de salida para cada línea de salida, una para los paquetes expeditos y otra para los paquetes regulares. Al llegar un paquete se pone en la cola correspondiente. La cola expedita tiene prioridad sobre la regular; por ejemplo, mediante el uso de un programador de prioridades. De esta forma, los paquetes expeditos ven una red sin carga, aun cuando, de hecho, haya una carga pesada de tráfico regular.

Reenvío asegurado

Hay un esquema un poco más elaborado para administrar las clases de servicios, el cual se conoce como **reenvío asegurado**. Se describe en el RFC 2597 y especifica que debe haber cuatro clases de prioridades, cada una con sus propios recursos. Las primeras tres clases se deben llamar oro, plata y bronce. Además, define tres probabilidades de descarte para los paquetes que están en congestión: baja, media y alta. En conjunto, estos dos factores definen 12 clases de servicios.

La figura 5-37 muestra una forma en que se pueden procesar los paquetes bajo el esquema de reenvío asegurado. El primer paso es clasificar los paquetes en una de las cuatro clases de prioridades. Como en el reenvío expedito, este paso se podría realizar en el host emisor (como se muestra en la figura) o en

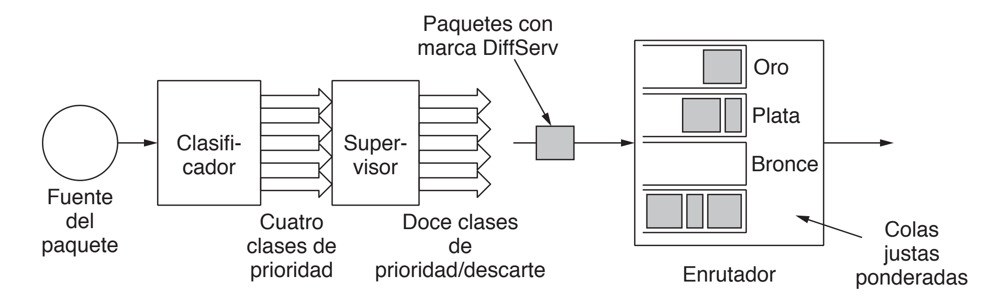


Figura 5-37. Una posible implementación del reenvío asegurado.

el enrutador de ingreso; el operador puede limitar la tasa de paquetes de prioridad más alta como parte del servicio ofrecido.

El siguiente paso es determinar la clase de descarte para cada paquete. Para ello es necesario pasar los paquetes de cada clase de prioridad por un supervisor de tráfico como una cubeta con tokens. El supervisor deja pasar todo el tráfico, pero identifica los paquetes que caben dentro de pequeñas ráfagas con un nivel de descarte bajo, a los paquetes que exceden las pequeñas ráfagas con un nivel de descarte medio y a los paquetes que exceden las ráfagas grandes con un nivel de descarte alto. Entonces, la combinación de las clases de prioridad y de descarte se codifica en cada paquete.

Por último, los enrutadores en la red procesan los paquetes mediante un programador de paquetes que distingue las diversas clases. Una elección común es usar el encolamiento justo ponderado para las cuatro clases de prioridad, en donde las clases superiores reciben pesos más altos. De esta forma, las clases superiores obtendrán la mayor parte del ancho de banda pero no se privará a las clases inferiores de todo el ancho de banda. Por ejemplo, si los pesos se duplican de una clase a la siguiente clase más alta, ésta tendrá el doble de ancho de banda. Dentro de una clase de prioridad, se puede dar preferencia a los paquetes con una clase de descarte más alta para descartarlos mediante la ejecución de un algoritmo tal como RED (Detección Temprana Aleatoria), que vimos en la sección 5.3.5. RED empezará a descartar paquetes a medida que se acumule la congestión, pero antes de que el enrutador se quede sin espacio en el búfer. En esta etapa aún hay espacio de búfer para aceptar los paquetes con bajo nivel de descarte mientras se descartan los paquetes con alto nivel de descarte.

5.5 INTERCONEXIÓN DE REDES

Hasta ahora hemos supuesto de manera implícita que hay una sola red homogénea, en donde cada máquina usa el mismo protocolo en cada capa. Por desgracia, este supuesto es demasiado optimista. Existen muchas redes distintas, entre ellas PAN, LAN, MAN y WAN. Ya describimos Ethernet, Internet por cable, las redes telefónicas fija y móvil, 802.11, 802.16 y muchas más. Hay numerosos protocolos con un uso muy difundido a través de estas redes, en cada capa. En las siguientes secciones estudiaremos con cuidado los aspectos que surgen cuando se conectan dos o más redes para formar una **interred** o simplemente una **internet**.

Sería más simple unir redes si todos usaran una sola tecnología de red; a menudo se da el caso de que hay un tipo dominante de red, como Ethernet. Algunos expertos especulan que la multiplicidad de tecnologías desaparecerá tan pronto como todos se den cuenta qué tan maravillosa es la red... No cuente con ello. La historia muestra que esto es sólo ilusión. Los distintos tipos de redes luchan con problemas diferentes; por ejemplo, las redes Ethernet y satelitales casi siempre difieren. Al reutilizar los sistemas existentes, como operar redes de datos a través del servicio de cable, la red telefónica y el cableado eléctrico, se agregan limitaciones que ocasionan la divergencia de las características de las redes. La heterogeneidad llegó para quedarse.

Si siempre habrá una variedad de redes, sería más simple que no necesitáramos interconectarlas. Esto también es muy poco probable. Bob Metcalfe postuló que el valor de una red con N nodos es el número de conexiones que se pueden hacer entre los nodos, o N^2 (Gilder, 1993). Esto significa que las redes grandes son mucho más valiosas que las pequeñas, ya que permiten muchas más conexiones, por lo que siempre habrá un incentivo para combinar redes pequeñas.

Internet es el ejemplo primordial de esta interconexión (vamos a escribir Internet con una "I" mayúscula para diferenciarla de otras interred o redes interconectadas). El propósito de unir todas estas redes es permitir que los usuarios en cualquiera de ellas se comuniquen con los usuarios de las otras redes. Cuando usted paga a un ISP por el servicio de Internet, el costo dependerá del ancho de banda de su línea, pero lo que en realidad está pagando es la capacidad de intercambiar paquetes con cualquier otro host que

también esté conectado a Internet. Después de todo, Internet no sería muy popular si sólo pudiéramos enviar paquetes a otros hosts en la misma ciudad.

Como por lo general las redes difrieren en formas importantes, no siempre es tan sencillo enviar paquetes de una red a otra. Debemos lidiar con problemas de heterogeneidad y también con problemas de escala a medida que la interred resultante aumenta su tamaño en forma considerable. Empezaremos por analizar la forma en que pueden diferir las redes, para ver a qué nos enfrentamos. Después veremos el método tan exitoso utilizado por IP (Protocolo de Internet), el protocolo de la capa de red de Internet, incluyendo las técnicas de tunelización a través de las redes, el enrutamiento en las interredes y la fragmentación de paquetes.

5.5.1 Cómo difieren las redes

Las redes pueden diferir de muchas maneras. Algunas de las diferencias, como técnicas de modulación o formatos de tramas diferentes, se encuentran en la capa física y en la de enlace de datos. No trataremos esas diferencias aquí. En su lugar, en la figura 5-38 listamos algunas diferencias que pueden ocurrir en la capa de red. La conciliación de estas diferencias es lo que hace más difícil la interconexión de redes que la operación con una sola red.

Cuando los paquetes enviados por una fuente en una red deben transitar a través de una o más redes foráneas antes de llegar a la red de destino, pueden ocurrir muchos problemas en las interfaces entre las redes. Para empezar, la fuente necesita dirigirse al destino. ¿Qué hacemos si la fuente está en una red Ethernet y el destino en una red WiMAX? Suponiendo que sea posible especificar un destino WiMAX desde una red Ethernet, los paquetes pasarían de una red sin conexión a una orientada a conexión. Para ello tal vez sea necesario establecer una conexión improvisada, lo cual introduce un retardo y mucha sobrecarga si la conexión no se utiliza para muchos paquetes más.

También debemos tener en cuenta muchas diferencias específicas. ¿Cómo enviamos un paquete mediante multidifusión a un grupo con ciertos miembros en una red que no soporta multidifusión? Los diferentes tamaños máximos de paquete usados por las diferentes redes también pueden ser una gran molestia. ¿Cómo se pasa un paquete de 8 000 bytes a través de una red cuyo tamaño máximo de paquete es de 1500 bytes? Si los paquetes en una red orientada a conexión transitan en una red sin conexión, pueden llegar en un orden distinto al que se enviaron. Esto es algo que el emisor probablemente no esperaba, y podría ser también una (desagradable) sorpresa para el receptor.

Aspecto	Algunas posibilidades
Servicio ofrecido.	Sin conexión *vs*. orientado a conexión.
Direccionamiento.	Distintos tamaños, plano o jerárquico.
Difusión.	Presente o ausente (también multidifusión).
Tamaño de paquete.	Cada red tiene su propio valor máximo.
Ordenamiento.	Entrega ordenada y desordenada.
Calidad del servicio.	Presente o ausente; muchos tipos distintos.
Confiabilidad.	Distintos niveles de pérdida.
Seguridad.	Reglas de privacidad, cifrado, etcétera.
Parámetros.	Distintos tiempos de expiración, especificaciones de flujo, etcétera..
Contabilidad.	Por tiempo de conexión, paquete, byte o ninguna.

Figura 5-38. Algunas de las diversas formas en que pueden diferir las redes.

Estos tipos de diferencias se pueden enmendar, con cierto esfuerzo. Por ejemplo, una puerta de enlace que une dos redes podría generar paquetes separados para cada destino en vez de un mejor soporte para la multidifusión. Un paquete extenso se podría dividir, enviar en piezas y unir de vuelta. Los receptores podrían colocar los paquetes en un búfer y entregarlos en orden.

Las redes también pueden diferir en sentidos más importantes que son más difíciles de conciliar. El ejemplo más claro es la calidad del servicio. Si una red tiene una QoS sólida y la otra ofrece un servicio del mejor esfuerzo, será imposible hacer garantías de ancho de banda y retardo para el tráfico en tiempo real de un extremo al otro. De hecho, es probable que sólo se puedan hacer mientras la red del mejor esfuerzo opere con poco uso, o cuando se utilice en raras ocasiones, lo cual no es muy probable que sea el objetivo de la mayoría de los ISP. Los mecanismos de seguridad son problemáticos, pero por lo menos el cifrado para la confiabilidad e integridad de los datos se puede poner encima de las redes que no lo incluyen de antemano. Por último, las diferencias en la contabilidad pueden producir facturas no deseadas cuando el uso normal de repente se vuelve costoso, como lo han descubierto los usuarios de teléfonos móviles cuando usan *roaming* y tienen planes de datos.

5.5.2 Cómo se pueden conectar las redes

Existen dos opciones básicas para conectar distintas redes: podemos construir dispositivos que traduzcan o conviertan los paquetes de cada tipo de red en paquetes para otra red o, como buenos científicos de la computación, podemos tratar de resolver el problema al agregar una capa de indirección y construir una capa común encima de las distintas redes. En cualquier caso, los dispositivos se colocan en los límites entre las redes.

Desde un principio, Cerf y Kahn (1974) abogaron por una capa común para ocultar las diferencias de las redes existentes. Esta metodología ha tenido un gran éxito, y la capa que propusieron se separó eventualmente en los protocolos TCP e IP. Casi cuatro décadas después, IP es la base de la Internet moderna. Por este logro, Cerf y Kahn recibieron el Premio Turing 2004, conocido de manera informal como el Premio Nobel de las ciencias computacionales. IP proporciona un formato de paquete universal que todos los enrutadores reconocen y que se puede pasar casi por cualquier red. IP ha extendido su alcance de las redes de computadoras para apoderarse de la red telefónica. También opera en redes de sensores y en otros dispositivos pequeños que alguna vez se consideraron con recursos demasiado restringidos como para soportarlo.

Hemos analizado varios dispositivos diferentes que conectan redes, incluyendo repetidores, hubs, switches, puentes, enrutadores y puertas de enlace. Los repetidores y hubs sólo desplazan bits de un cable a otro. En su mayoría son dispositivos analógicos y no comprenden nada sobre los protocolos de las capas superiores. Los puentes y switches operan en la capa de enlace. Se pueden usar para construir redes, pero sólo con una pequeña traducción de protocolos en el proceso; por ejemplo, entre los switches Ethernet de 10, 100 y 1000 Mbps. Nuestro enfoque en esta sección es en los dispositivos de interconexión que operan en la capa de red, es decir, los enrutadores. Dejaremos las puertas de enlace, que son dispositivos de interconexión de las capas superiores, para después.

Primero exploraremos a un alto nivel la forma en que se puede usar la interconexión con una capa de red común para interconectar redes distintas. En la figura 5-39(a) se muestra una interred compuesta por redes 802.11, MPLS y Ethernet. Suponga que la máquina fuente en la red 802.11 desea enviar un paquete a la máquina de destino en la red Ethernet. Como estas tecnologías son distintas y además están separadas por otro tipo de red (MPLS), se requiere un procesamiento adicional en los límites entre las redes.

En general, como dferentes redes pueden tener distintas formas de direccionamiento, el paquete transporta una dirección de capa de red que puede identificar cualquier host a través de las tres redes.

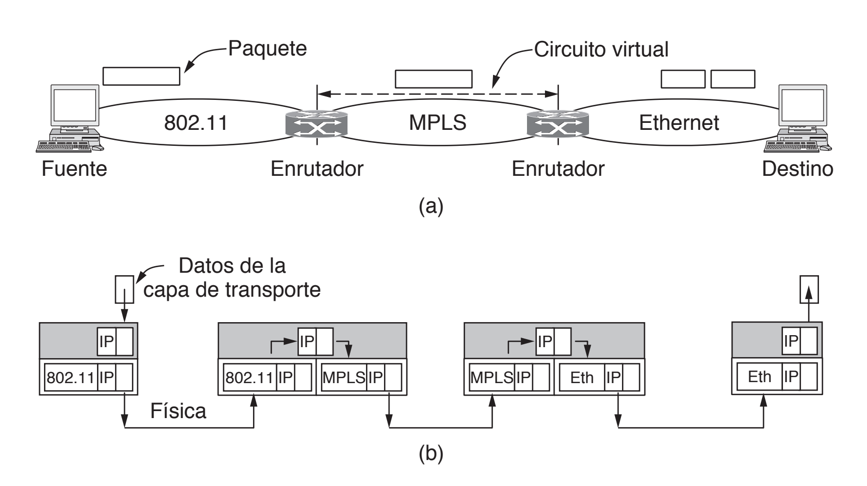


Figura 5-39. (a) Un paquete que cruza distintas redes. (b) Procesamiento de protocolos de las capas de red y de enlace.

El primer límite al que llega el paquete es cuando cambia de una red 802.11 a una red MPLS. 802.11 proporciona un servicio sin conexión, pero MPLS provee un servicio orientado a conexión. Esto significa que se debe establecer un circuito virtual para cruzar esa red. Una vez que el paquete viaje por el circuito virtual, llegará a la red Ethernet. En este límite tal vez el paquete sea demasiado grande como para transportarlo, ya que 802.11 puede trabajar con tramas más grandes que Ethernet. Para manejar este problema, el paquete se divide en fragmentos y cada fragmento se envía por separado. Cuando los fragmentos llegan a su destino, se vuelven a ensamblar. Entonces es cuando el paquete ha completado su viaje.

En la figura 5-39(b) se muestra el procesamiento de los protocolos para este viaje. La fuente acepta los datos de la capa de transporte y genera un paquete con el encabezado de la capa de red común, que en este ejemplo es IP. El encabezado de red contiene la dirección de destino final, que se utiliza para determinar que se debe enviar el paquete a través del primer enrutador. Así, el paquete se encapsula en una trama 802.11, cuyo destino es el primer enrutador, y se transmite. En el enrutador, el paquete se extrae del campo de datos de la trama y se descarta el encabezado de la trama 802.11. Ahora el enrutador examina la dirección IP en el paquete y busca esta dirección en su tabla de enrutamiento. Con base en esta dirección, decide enviar a continuación el paquete al segundo enrutador. Para esta parte de la ruta, es necesario establecer un circuito virtual MPLS hacia el segundo enrutador y encapsular el paquete con encabezados MPLS que viajen por este circuito. En el extremo lejano, se descarta el encabezado MPLS y de nuevo se consulta la dirección de red para buscar el siguiente salto en la capa de red, el destino en sí. Como el paquete es demasiado largo para enviarlo a través de Ethernet, se divide en dos partes. Cada una se coloca en el campo de datos de una trama Ethernet y se envía a la dirección Ethernet del destino. Ya en el destino, se elimina el encabezado Ethernet de cada una de las tramas y se vuelve a ensamblar el contenido. Finalmente el paquete ha llegado a su destino.

Cabe mencionar que hay una diferencia esencial entre el caso de enrutamiento y el caso de conmutación (o puenteo). Con un enrutador, el paquete se extrae de la trama y la dirección de red en el paquete se utiliza para decidir a dónde enviarlo. Con un switch (o puente), toda la trama se transporta con base en su dirección MAC. Los switches no tienen que entender el protocolo de capa de red que se utiliza para conmutar los paquetes. Los enrutadores sí tienen que hacerlo.

Por desgracia, la interconexión de redes no es tan fácil como suena. De hecho, cuando se introdujeron los puentes, la intención era que unieran distintos tipos de redes, o por lo menos distintos tipos de LAN. Para ello debían traducir tramas de una LAN en tramas de otra LAN. Sin embargo, esto no funcionó bien debido a la misma razón por la que es difícil la interconexión de redes: las diferencias en las características de las LAN (como los distintos tamaños máximos de los paquetes, y las LAN con y sin clases de prioridad) son difíciles de enmascarar. En la actualidad, el uso más común de los puentes es para conectar el mismo tipo de red en la capa de enlace, y los enrutadores conectan distintas redes en la capa de red.

La interconexión de redes ha tenido mucho éxito en la creación de redes extensas, pero sólo funciona cuando hay una capa de red común. De hecho, con el tiempo han surgido varios protocolos de red. Es difícil lograr que todos estén de acuerdo en cuanto a un solo formato cuando las empresas perciben como ventaja comercial el tener un formato propietario que puedan controlar. Algunos ejemplos además de IP, que ahora es el protocolo de red casi universal, son: IPX, SNA y AppleTalk. Ninguno de estos protocolos se utiliza mucho en la actualidad, pero siempre habrá otros protocolos. Probablemente ahora el ejemplo más relevante sea el de IPv4 e IPv6. Aunque ambas son versiones de IP, no son compatibles (o no hubiera sido necesario crear IPv6).

Un enrutador que puede manejar múltiples protocolos de red se denomina **enrutador multiprotocolo**. Éste debe traducir los protocolos o dejar una conexión para una capa de protocolo superior. Ninguna de las dos opciones es totalmente satisfactoria. Para una conexión en una capa superior (por decir, mediante el uso de TCP) se requiere que todas las redes implementen TCP (lo cual tal vez no sea posible). En consecuencia, se limita el uso en las redes a las aplicaciones que usan TCP (lo cual no incluye a muchas aplicaciones en tiempo real).

La alternativa es traducir los paquetes entre las redes. Sin embargo, a menos que los formatos de los paquetes sean muy similares y tengan los mismos campos de información, tales conversiones siempre serán incompletas y con frecuencia estarán destinadas al fracaso. Por ejemplo, las direcciones IPv6 son de 128 bits de longitud. No cabrán en un campo de dirección IPv4 de 32 bits, sin importar qué tanto se esfuerce el enrutador. El hecho de lograr que IPv4 e IPv6 funcionen en la misma red, ha demostrado ser un gran obstáculo para la implementación de IPv6 (para ser justos, también ha sido muy difícil hacer que los clientes entiendan por qué les convendría usar IPv6 en primer lugar). Se pueden esperar mayores problemas al traducir entre protocolos básicamente distintos, como los protocolos de red sin conexión y los orientados a conexión. Dadas estas dificultades, la conversión sólo se intenta raras veces. Sin duda, la razón por la que IP ha funcionado tan bien se debe a que sirve como un tipo de mínimo común denominador. Requiere muy poco de las redes en las que opera, pero ofrece como resultado sólo un servicio del mejor esfuerzo.

5.5.3 Tunelización

Es en extremo difícil manejar el caso general de hacer que dos redes distintas se interconecten. Sin embargo, hay un caso especial que se puede manejar, incluso para distintos protocolos de red. Este caso es cuando el host de origen y el de destino están en el mismo tipo de red, pero hay una red diferente en medio. Como ejemplo, piense en un banco internacional con una red IPv6 en París, una en Londres y una conectividad entre las oficinas a través de la Internet IPv4. Esta situación se muestra en la figura 5-40.

La solución a este problema es una técnica llamada **tunelización** (*tunneling*). Para enviar un paquete IP a un host en la oficina de Londres, un host en la oficina de París construye el paquete que contiene una dirección IPv6 en Londres y la envía al enrutador multiprotocolo que conecta la red IPv6 de París con la Internet IPv4. Cuando este enrutador recibe el paquete IPv6, lo encapsula con un encabezado IPv4

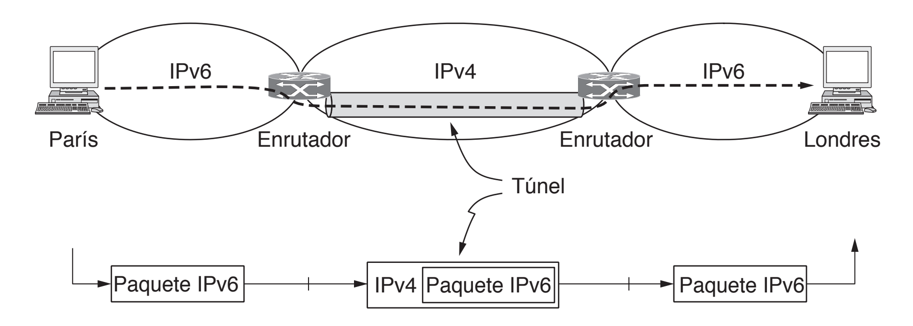


Figura 5-40. Tunelización de un paquete de París a Londres.

dirigido al lado IPv4 del enrutador multiprotocolo que se conecta con la red IPv6 de Londres. Es decir, el enrutador coloca un paquete (IPv6) dentro de un paquete (IPv4). Cuando llega este paquete envuelto, el enrutador de Londres extrae el paquete IPv6 original y lo envía hacia el host de destino.

Podemos visualizar la ruta hacia la Internet IPv4 como un gran túnel que se extiende de un enrutador multiprotocolo al otro. El paquete IPv6 simplemente viaja de un extremo del túnel al otro, bien acomodado en una caja bonita. No tiene que preocuparse por lidiar con IPv4 para nada. Tampoco tienen que hacerlo los hosts en París o en Londres. Sólo los enrutadores multiprotocolo tienen que entender los paquetes IPv4 e IPv6. De hecho, el recorrido completo desde un enrutador multiprotocolo al otro es como un salto a través de un solo enlace.

Podemos usar una analogía para aclarar el concepto de tunelización. Considere una persona que maneja su auto de París a Londres. En Francia, el auto se mueve con su propia energía, pero al llegar al Canal de la Mancha, se carga en un tren de alta velocidad y se transporta a Inglaterra a través del Eurotúnel (no se pueden conducir autos a través del Eurotúnel). En efecto, el auto se transporta como carga, como se muestra en la figura 5-41. En el otro extremo se libera el auto en las carreteras inglesas y se vuelve a mover por sus propios medios. La tunelización de paquetes a través de una red foránea funciona de la misma manera.

La tunelización se utiliza mucho para conectar hosts y redes aisladas mediante el uso de otras redes. La red que resulta se denomina **red superpuesta** (*overlay*), ya que realmente está superpuesta sobre la red base. Implementar un protocolo de red con una nueva característica es un motivo común, como se indica en nuestro ejemplo de “IPv6 sobre IPv4”. La desventaja de la tunelización es que no se puede llegar a ninguno de los hosts en la red que se tuneliza debido a que los paquetes no pueden escapar a mitad del túnel. Sin embargo, esta limitación de los túneles se convierte en una ventaja gracias a las redes **VPN** (**Redes Privadas Virtuales**, del inglés *Virtual Private Network*). Una VPN es simplemente una red superpuesta que se utiliza para proporcionar una medida de seguridad. Exploraremos las VPN cuando lleguemos al capítulo 8.

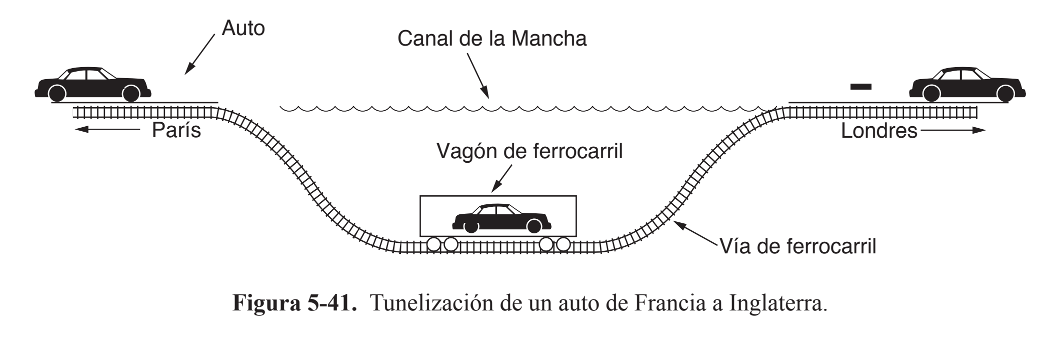


Figura 5-41. Tunelización de un auto de Francia a Inglaterra.

5.5.4 Enrutamiento entre redes

El enrutamiento a través de una interred presenta el mismo problema que el enrutamiento en una sola red, pero con algunas complicaciones adicionales. Para empezar, las redes pueden usar internamente distintos algoritmos de enrutamiento. Por ejemplo, una red podría usar un enrutamiento por estado del enlace y otra un enrutamiento por vector de distancia. Como los algoritmos de estado del enlace necesitan conocer la topología pero los algoritmos de vector de distancia no, tan sólo esta diferencia dificultaría mucho el proceso de encontrar las rutas más cortas a través de la internet.

Las redes manejadas por distintos operadores conducen a problemas más grandes. En primer lugar, tal vez los operadores tengan distintas ideas sobre lo que sería una buena ruta a través de la red. Posiblemente un operador quiera la ruta con el menor retardo, mientras que otro prefiera la ruta menos costosa. Esto obligará a los operadores a usar distintas cantidades para establecer los costos de la ruta más corta (por ejemplo, milisegundos de retardo en comparación con el costo monetario). Como no se compararán los pesos entre las redes, las rutas más cortas en la interred no estarán bien definidas.

O peor aún, tal vez un operador no quiera que otro conozca siquiera los detalles de las rutas en su red, quizá debido a que los pesos y las rutas puedan reflejar información delicada (como el costo monetario) que representa una ventaja comercial competitiva.

Por último, la interred puede ser mucho más grande que cualquiera de las redes que la conforman. Por lo tanto, puede requerir algoritmos de enrutamiento que escalen bien mediante el uso de una jerarquía, incluso si ninguna de las redes individuales necesita usar una jerarquía.

Todas estas consideraciones conducen a un algoritmo de enrutamiento de dos niveles. Dentro de cada red, se utiliza un **protocolo intradominio** o de **puerta de enlace interior** para el enrutamiento ("puerta de enlace" es un término antiguo para "enrutador"). Podría ser un protocolo de estado del enlace del tipo que ya hemos descrito. A través de las redes que conforman la interred se utiliza un **protocolo interdominio** o de **puerta de enlace exterior**. Todas las redes pueden usar distintos protocolos intradominio, pero deben usar el mismo protocolo interdominio. En Internet, el protocolo de enrutamiento interdominio se denomina **BGP** (**Protocolo de Puerta de Enlace de Frontera**, del inglés *Border Gateway Protocol*). Lo describiremos en la siguiente sección.

Hay un término importante más que debemos introducir. Como cada red se opera de manera independiente a las demás, se conoce comúnmente como un **AS** (**Sistema Autónomo**, del inglés *Autonomous System*). Una red de un ISP es un buen modelo mental de un AS. De hecho, una red de ISP puede estar compuesta por más de un AS en caso de ser administrada, o, si ha sido adquirida, por múltiples redes. Pero en general, la diferencia no es importante.

Por lo común, los dos niveles no son estrictamente jerárquicos, pues se podrían generar rutas muy subóptimas si una red internacional extensa y una red regional pequeña se abstrajeran para formar una sola red. Sin embargo, en realidad se expone muy poca información sobre las rutas dentro de la red para encontrar rutas a través de la interred. Esto ayuda a lidiar con todas las complicaciones. Mejora el escalamiento y permite a los operadores elegir libremente las rutas dentro de sus propias redes, mediante el uso de un protocolo de su elección. Además, tampoco se tienen que comparar los pesos a través de las redes ni exponer la información delicada fuera de ellas.

No obstante, hemos dicho muy poco hasta ahora sobre la forma en que se determinan las rutas a través de las redes de la interred. En Internet, un gran factor determinante consiste en los arreglos comerciales entre los ISP. Cada ISP puede cobrar o recibir dinero de los otros ISP por transportar tráfico. Otro factor es que, si el enrutamiento de la interred requiere cruzar límites internacionales, de repente pueden entrar en juego varias leyes, como las estrictas leyes de privacidad de Suecia con relación a la exportación de datos personales sobre los ciudadanos suecos. Todos estos factores no técnicos están envueltos en el concepto de una **política de enrutamiento** que gobierna la forma en que las redes autónomas seleccionan los enrutadores que van a usar. Volveremos a las políticas de enrutamiento cuando hablemos sobre el BGP.

5.5.5 Fragmentación de paquetes

Cada red o enlace impone un tamaño máximo a sus paquetes. Estos límites tienen varias razones, entre ellas:

1. El hardware (por ejemplo, el tamaño de una trama Ethernet).
2. El sistema operativo (por ejemplo, todos los búferes son de 512 bytes).
3. Los protocolos (por ejemplo, la cantidad de bits en el campo de longitud de paquete).
4. El cumplimiento de algún estándar (inter)nacional.
5. El deseo de reducir hasta cierto nivel las retransmisiones inducidas por errores.
6. El deseo de evitar que un paquete ocupe el canal demasiado tiempo.

El resultado de todos estos factores es que los diseñadores de redes no tienen la libertad de elegir cualquier tamaño máximo de paquetes que deseen. Las cargas útiles máximas para algunas tecnologías comunes son de 1500 bytes para Ethernet y 2 272 para 802.11. El protocolo IP es más generoso, ya que permite paquetes de hasta 65 515 bytes.

Por lo general, los hosts prefieren transmitir paquetes grandes, ya que esto reduce las sobrecargas de paquetes, como el ancho de banda desperdiciado en los bytes de encabezado. Surge un problema obvio de interconexión de redes cuando un paquete grande quiere viajar a través de una red cuyo tamaño máximo de paquete es muy pequeño. Esta molestia ha sido una cuestión persistente; y las soluciones han evolucionado junto con toda la experiencia que se ha obtenido gracias a Internet.

Una solución es asegurar que no ocurra el problema en primer lugar. Sin embargo, es más fácil decir esto que hacerlo. Por lo general, una fuente no conoce la ruta que tomará un paquete a través de la red hacia un destino, por lo que en definitiva no sabe qué tan pequeños deben ser los paquetes para llegar ahí. A este tamaño de paquete se le conoce como **MTU de la ruta** (**Unidad de Transmisión Máxima de la ruta**, del ingés *Path Maximum Transmission Unit*). Incluso si la fuente conociera el MTU de la ruta, los paquetes se enrutan de manera independiente en una red sin conexión como Internet. Este enrutamiento significa que las rutas pueden cambiar de repente, lo que a su vez puede cambiar de manera inesperada el MTU de la ruta.

La solución alternativa al problema es permitir que los enrutadores dividan los paquetes en **fragmentos** y envíen cada uno como un paquete de red separado. Sin embargo, como lo sabe cualquier padre de un niño pequeño, es mucho más fácil convertir un objeto grande en pequeños fragmentos que el proceso inverso (los físicos incluso le han dado un nombre a este efecto: Segunda Ley de la termodinámica). Las redes de conmutación de paquetes también tienen problemas al unir nuevamente los fragmentos.

Existen dos estrategias opuestas para recombinar los fragmentos de vuelta en el paquete original. La primera es hacer que la fragmentación producida por una red de "pequeños paquetes" sea transparente para cualquier red subsecuente por la que deba pasar el paquete en su camino hacia el destino final. Esta opción se muestra en la figura 5-42(a). En este método, cuando un paquete de tamaño excesivo llega a *G1*, el enrutador lo divide en fragmentos. Cada fragmento es dirigido al mismo enrutador de salida, *G2*, en donde se recombinan las piezas. De esta manera se ha hecho transparente el paso a través de la red de paquete pequeño. Las redes subsecuentes ni siquiera se enteran de que ha ocurrido una fragmentación.

La fragmentación transparente es sencilla, pero tiene algunos problemas. Por una parte, el enrutador de salida debe saber cuándo ha recibido todas las piezas, por lo que debe incluirse un campo de conteo o un bit de "fin de paquete". Además, como todos los paquetes deben salir por el mismo enrutador para volver a ensamblarlos, las rutas están restringidas. Al no permitir que algunos fragmentos sigan una ruta al destino final, y otros fragmentos una ruta distinta, puede bajar un poco el desempeño. Lo más importante es la cantidad de trabajo que el enrutador tenga que llevar a cabo. Tal vez necesite colocar los fragmentos

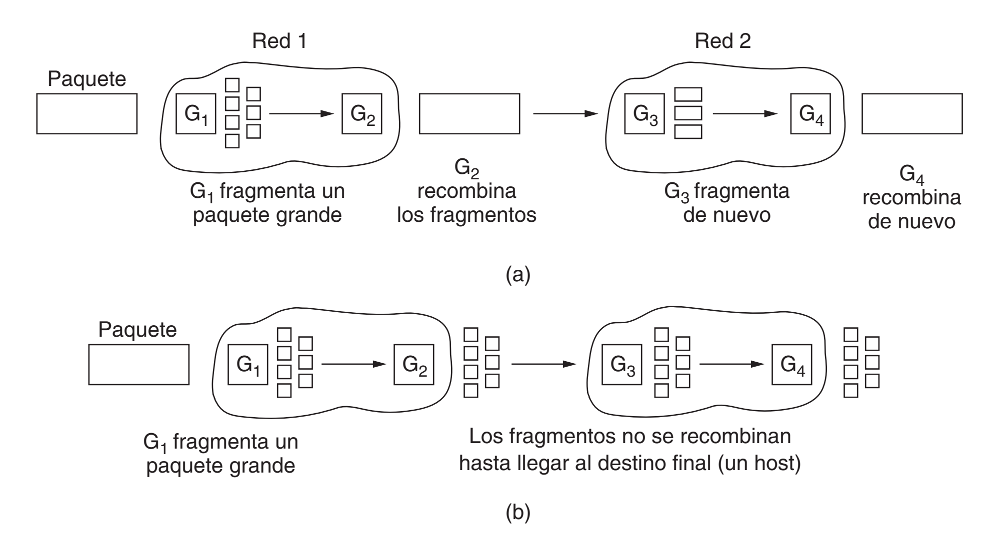


Figura 5-42. (a) Fragmentación transparente. (b) Fragmentación no transparente.

en un búfer a medida que vayan llegando, para después decidir cuándo debe descartarlos si no llegan todos los fragmentos. Parte de este trabajo también podría ser un desperdicio, ya que el paquete puede pasar a través de una serie de pequeñas redes de paquetes, y tal vez haya que fragmentarlo y recombinarlo repetidas veces.

La otra estrategia de fragmentación es abstenerse de recombinar los fragmentos en los enrutadores intermedios. Una vez que se ha fragmentado un paquete, cada fragmento se trata como si fuera un paquete original. Los enrutadores pasan los fragmentos, como se muestra en la figura 5-42(b); la recombinación ocurre sólo en el host de destino.

La principal ventaja de la fragmentación no transparente es que los enrutadores tienen que trabajar menos. IP funciona de esta manera. Un diseño completo requiere que los fragmentos se enumeren de tal forma que se pueda reconstruir el flujo de datos original. El diseño utilizado por IP es proporcionar a cada fragmento un número de paquete (que todos los paquetes transportan), un desplazamiento de bytes absoluto dentro del paquete, y una bandera que indica si es el final del paquete. En la figura 5-43 se muestra un ejemplo. Aunque es simple, este diseño tiene algunas propiedades atractivas. Los fragmentos se pueden colocar en un búfer en el destino, en el lugar apropiado para recombinarlos, aun cuando lleguen desordenados. Los fragmentos también se pueden fragmentar si pasan a través de una red con una MTU aún más pequeña. Esto se muestra en la figura 5-43(c). Las retransmisiones del paquete (si no se recibieron todos los fragmentos) se pueden fragmentar en distintas piezas. Por último, los fragmentos pueden ser de un tamaño arbitrario, incluso hasta de un solo byte más el encabezado del paquete. En todos los casos, el destino simplemente usa el número de paquete y el desplazamiento del fragmento para colocar los datos en la posición correcta, y la bandera de fin de paquete para determinar cuándo tiene el paquete completo.

Por desgracia, este diseño aún tiene problemas. La sobrecarga puede ser mayor que con la fragmentación transparente, ya que los encabezados de los fragmentos ahora se transportan a través de algunos enlaces en donde tal vez no sean necesarios. Pero el verdadero problema es la existencia de los fragmentos en primera instancia. Kent y Mogul (1987) argumentaron que la fragmentación es perjudicial para el desempeño, ya que al igual que las sobrecargas de los encabezados, un paquete completo se extravía si cualquiera de sus fragmentos se pierde, y porque la fragmentación representa una carga para los hosts más grande de lo que se tenía pensado originalmente.

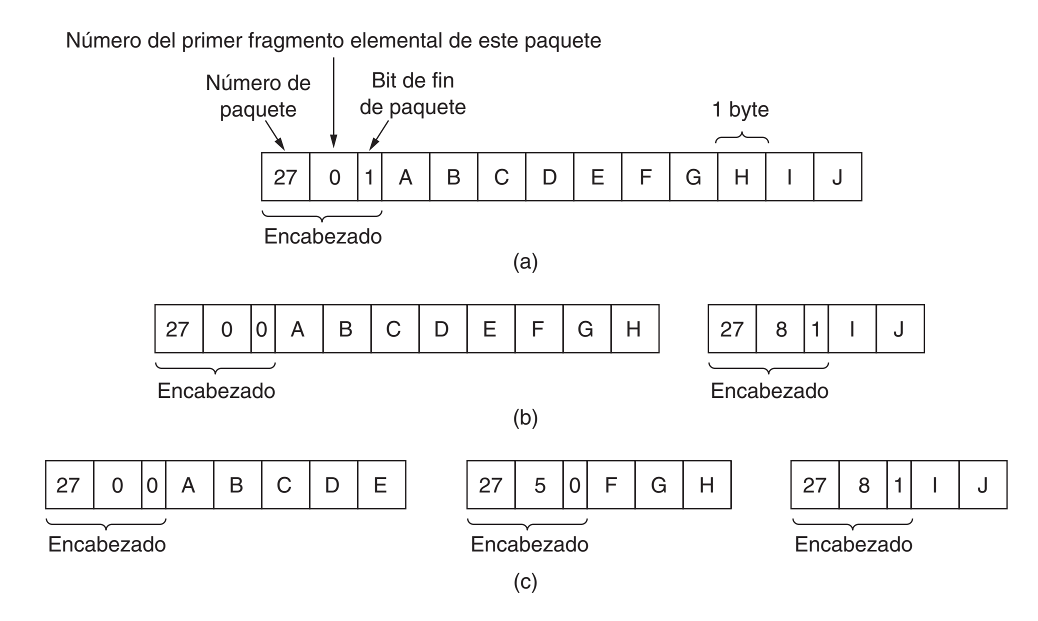


Figura 5-43. La fragmentación cuando el tamaño de datos elemental es de 1 byte. (a) El paquete original que contiene 10 bytes de datos. (b) Los fragmentos después de pasar por una red con un tamaño máximo de paquete de 8 bytes de carga útil más encabezado. (c) Fragmentos después de pasar a través de una puerta de enlace de tamaño 5.

Esto nos conduce de vuelta a la solución original de deshacernos de la fragmentación en la red, la estrategia que se utiliza en la Internet moderna. Al proceso se le conoce como **descubrimiento de MTU de la ruta** (Mogul y Deering, 1990) y trabaja de la siguiente manera. Cada paquete IP se envía con sus bits de encabezado establecidos para indicar que no se puede realizar ningún tipo de fragmentación. Si un enrutador recibe un paquete demasiado grande, genera un paquete de error, lo devuelve a la fuente y descarta el paquete. Esto se muestra en la figura 5-44. Cuando el origen recibe el paquete de error, usa la información interna para volver a fragmentar el paquete en piezas que sean lo bastante pequeñas como para que el enrutador las pueda manejar. Si un enrutador más adelante en la ruta tiene una MTU aún más pequeña, se repite el proceso.

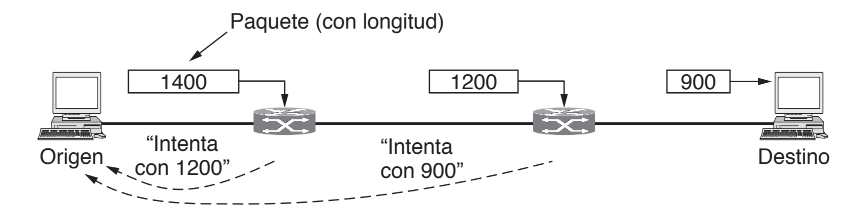


Figura 5-44. Descubrimiento de MTU de la ruta.

La ventaja del descubrimiento de MTU de la ruta es que ahora la fuente sabe la longitud del paquete que puede enviar. Si las rutas y la MTU de la ruta cambian, se activarán nuevos paquetes de error y la fuente se adaptará a la nueva ruta. Sin embargo, la fragmentación sigue siendo necesaria entre la fuente y el destino, a menos que las capas superiores descubran la MTU de la ruta y pasen la cantidad correcta de datos al protocolo IP. Por lo general, TCP e IP se implementan en conjunto (como "TCP/IP") para pasar

este tipo de información. Aun si esto no se hace con otros protocolos, de todas formas la fragmentación se sacó de la red y se pasó a los hosts.

La desventaja del descubrimiento de MTU de la ruta es que puede haber retardos iniciales adicionales sólo por enviar un paquete. Tal vez sea necesario más de un retardo de ida y vuelta para sondear la ruta y encontrar la MTU antes de entregar datos al destino. Aquí surge la pregunta acerca de si hay mejores diseños. Quizá la respuesta sea "sí". Considere el diseño en el que cada enrutador simplemente trunca los paquetes que exceden su MTU. Esto aseguraría que el destino conozca la MTU lo más rápido posible (con base en la cantidad de datos entregados) y reciba algunos de los datos.

5.6 LA CAPA DE RED DE INTERNET

Ahora es momento de examinar a conciencia la capa de red de Internet. Pero antes de entrar en detalles, vale la pena dar un vistazo a los principios que guiaron su diseño en el pasado y que hicieron posible el éxito que tiene hoy en día. En la actualidad, con frecuencia parece que la gente los ha olvidado. Estos principios se enumeran y analizan en el RFC 1958, el cual vale la pena leer (y debería ser obligatorio para todos los diseñadores de protocolos, así como un examen al final de la lectura). Este RFC utiliza mucho las ideas establecidas por Clark (1988) y por Saltzer y colaboradores (1984). Ahora sintetizaremos los que consideramos como los 10 mejores principios (del más al menos importante).

1. **Asegurarse de que funciona.** No termine el diseño o estándar hasta que múltiples prototipos se hayan comunicado entre sí de manera exitosa. Con mucha frecuencia, los diseñadores primero escriben un estándar de 1000 páginas, logran que se apruebe y después descubren que tiene demasiadas fallas y no funciona.Entonces escriben una versión 1.1 del estándar. Ésta no es la manera correcta de proceder.
2. **Mantener la simplicidad.** Cuando tenga duda, utilice la solución más simple. William de Occam formuló este principio (la navaja de Occam) en el siglo XIV. Dicho en términos modernos: combata las características. Si una característica no es absolutamente esencial, descártela, especialmente si el mismo efecto se puede alcanzar mediante la combinación de otras características.
3. **Elegir opciones claras.** Si hay varias maneras para realizar la misma tarea, elija sólo una. Tener dos o más formas de hacer lo mismo es buscarse problemas. Con frecuencia, los estándares tienen múltiples opciones, modos o parámetros debido a que personas poderosas insisten en que su método es el mejor. Los diseñadores deben resistir con fuerza esta tendencia. Simplemente diga no.
4. **Explotar la modularidad.** Este principio lleva directamente a la idea de tener pilas de protocolos, cuyas capas sean independientes entre sí. De esta forma, si las circunstancias requieren que un módulo o capa cambie, los otros no se verán afectados.
5. **Prevenir la heterogeneidad.** En cualquier red grande habrán diferentes tipos de hardware, facilidades de transmisión y aplicaciones. Para manejarlos, el diseño de la red debe ser simple, general y flexible.
6. **Evitar las opciones y parámetros estáticos.** Si los parámetros son inevitables (por ejemplo, el tamaño máximo del paquete), es mejor hacer que el emisor y el receptor negocien un valor que definir opciones fijas.
7. **Buscar un buen diseño no es necesario que sea perfecto.** Con frecuencia, los diseñadores tienen un buen diseño pero éste no puede manejar algún caso especial. En lugar de desbaratar el diseño, los diseñadores deberían optar por el buen diseño y dejar que esa parte la resuelvan las personas con requisitos extraños.

8. **Ser estricto cuando envíe y tolerante cuando reciba.** En otras palabras, sólo envíe paquetes que cumplan rigurosamente con los estándares, pero espere paquetes que tal vez no cumplan del todo y trate de lidiar con ellos.
9. **Pensar en la escalabilidad.** Si el sistema debe manejar de manera efectiva millones de hosts y miles de millones de usuarios, no se toleran bases de datos centralizadas de ningún tipo y la carga se debe dispersar de la manera más equitativa posible entre los recursos disponibles.
10. **Considerar el desempeño y el costo.** Si una red tiene un desempeño pobre o un costo exagerado, nadie la utilizará.

Dejemos a un lado los principios generales y comencemos a ver los detalles de la capa de red de Internet. En la capa de red, la Internet puede verse como un conjunto de redes, o **Sistemas Autónomos** (**AS**) interconectados. No hay una estructura real, pero existen varias redes troncales (*backbones*) principales. Éstas se construyen a partir de líneas de alto ancho de banda y enrutadores rápidos. Las más grandes de estas redes troncales, a la que se conectan todos los demás para llegar al resto de Internet, se llaman **redes de Nivel 1**. Conectadas a las redes troncales hay ISP (Proveedores de Servicio de Internet) que proporcionan acceso a Internet para los hogares y negocios, centros de datos e instalaciones de colocación llenas de máquinas servidores y redes regionales (de nivel medio). Los centros de datos sirven gran parte del contenido que se envía a través de Internet. A las redes regionales están conectados más ISP, LAN de muchas universidades y empresas, y otras redes de punta. En la figura 5-45 se presenta un dibujo de esta organización cuasijerárquica.

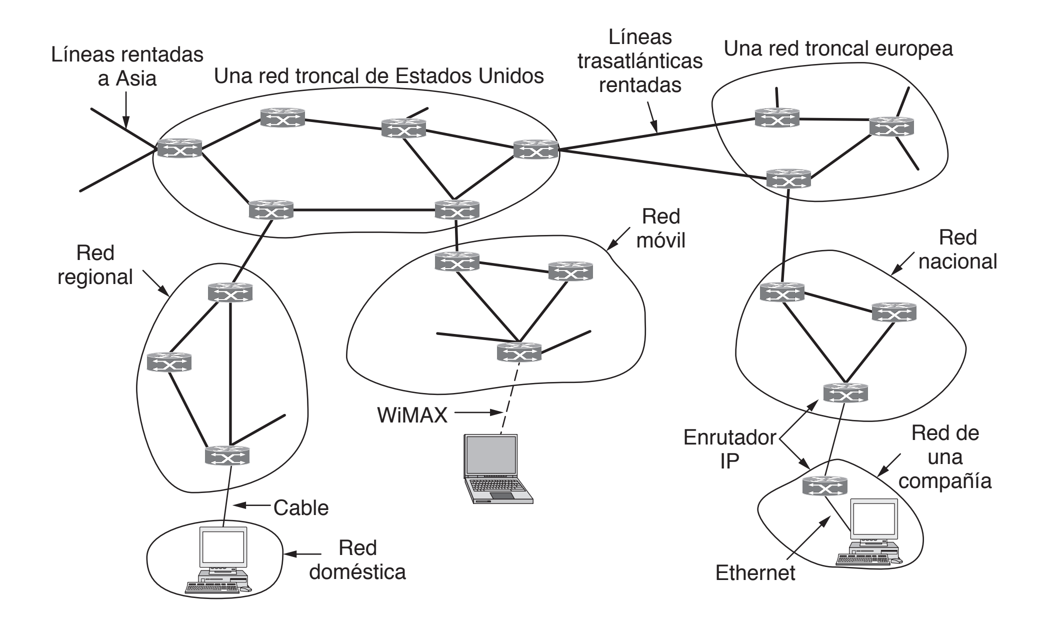


Figura 5-45. Internet es una colección interconectada de muchas redes.

El pegamento que mantiene unida a Internet es el protocolo de capa de red, **IP** (**Protocolo de Internet,** del inglés *Internet Protocol*). A diferencia de la mayoría de los protocolos de capa de red anteriores, IP se diseñó desde el principio con la interconexión de redes en mente. Una buena manera de visualizar la capa de red es la siguiente: su trabajo es proporcionar un medio de mejor esfuerzo (es decir, sin garantía) para transportar paquetes de la fuente al destino, sin importar si estas máquinas están en la misma red o si hay otras redes entre ellas.

La comunicación en Internet funciona de la siguiente manera. La capa de transporte toma flujos de datos y los divide para poder enviarlos como paquetes IP. En teoría, los paquetes pueden ser de hasta 64 Kbytes cada uno, pero en la práctica por lo general no sobrepasan los 1500 bytes (ya que son colocados en una trama de Ethernet). Los enrutadores IP reenvían cada paquete a través de Internet, a lo largo de una ruta de un enrutador a otro, hasta llegar al destino. En el destino, la capa de red entrega los datos a la capa de transporte, que a su vez los entrega al proceso receptor. Cuando todas las piezas llegan finalmente a la máquina de destino, la capa de red las vuelve a ensamblar para formar el datagrama original. A continuación este datagrama se entrega a la capa de transporte.

En el ejemplo de la figura 5-45, un paquete que se origina en la red doméstica tiene que atravesar cuatro redes y muchos enrutadores IP antes de llegar a la red de la compañía en la que se encuentra el host de destino. Esto es muy común en la práctica e incluso hay varias rutas más largas. También hay mucha conectividad redundante en Internet, con redes troncales e ISP conectados entre sí en varias ubicaciones. Esto significa que hay muchas rutas posibles entre dos hosts. Los protocolos de enrutamiento IP tienen la tarea de decidir qué rutas usar.

5.6.1 El protocolo IP versión 4

Un lugar adecuado para comenzar nuestro estudio de la capa de red de Internet es el formato de los datagramas de IP mismos. Un datagrama IPv4 consiste en dos partes: el encabezado y el cuerpo o carga útil. El encabezado tiene una parte fija de 20 bytes y una parte opcional de longitud variable. El formato del encabezado se muestra en la figura 5-46. Los bits se transmiten en orden de izquierda a derecha y de arriba hacia abajo, comenzando por el bit de mayor orden del campo *Versión* (éste es un orden de bytes de red *big endian*. En las máquinas *little endian*, como las computadoras Intel x86, se requiere una conversión por software tanto para la transmisión como para la recepción). En retrospectiva, el formato *little endian* hubiera sido una mejor opción, pero al momento de diseñar IP nadie sabía que llegaría a dominar la computación.

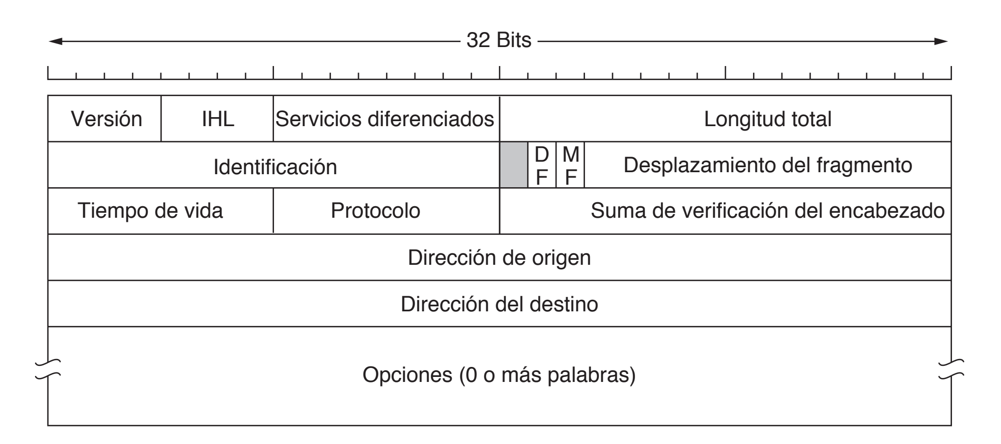


Figura 5-46. El encabezado de IPv4 (Protocolo de Internet).

El campo *Versión* lleva el registro de la versión del protocolo al que pertenece el datagrama. La versión 4 es la que domina Internet en la actualidad, y ahí es donde empezamos nuestro estudio. Al incluir la versión al inicio de cada datagrama, es posible tener una transición entre versiones a través de un largo periodo de tiempo. De hecho, IPv6 (la siguiente versión de IP) se definió hace más de una

década y apenas se está empezando a implementar. Lo describiremos más adelante en esta sección. En un momento dado nos veremos obligados a usarlo cuando cada uno de los casi 2^{31} habitantes de China tenga una PC de escritorio, una computadora portátil y un teléfono IP. Como observación adicional en cuanto a la numeración, IPv5 fue un protocolo de flujo experimental en tiempo real que nunca fue muy popular.

Dado que la longitud del encabezado no es constante, se incluye un campo en el encabezado (*IHL*) para indicar su longitud en palabras de 32 bits. El valor mínimo es de 5, cifra que se aplica cuando no hay opciones. El valor máximo de este campo de 4 bits es 15, lo que limita el encabezado a 60 bytes y por lo tanto, el campo *Opciones* a 40 bytes. Para algunas opciones, por ejemplo una que registre la ruta que ha seguido un paquete, 40 bytes es muy poco, lo que hace inútiles estas opciones.

El campo de *Servicio diferenciado* es uno de los pocos campos que ha cambiado su significado (ligeramente) con el transcurso de los años. En un principio el nombre de este campo era *Tipo de servicio*. Su propósito era, y aún es, distinguir entre las diferentes clases de servicios. Son posibles varias combinaciones de confiabilidad y velocidad. Para voz digitalizada, la entrega rápida le gana a la entrega precisa. Para la transferencia de archivos, es más importante la transmisión libre de errores que la transmisión rápida. El campo *Tipo de servicio* contaba con 3 bits para indicar la prioridad y 3 bits para indicar si a un host le preocupaba más el retardo, la velocidad de transmisión real o la confiabilidad. Sin embargo, nadie sabía realmente qué hacer con estos bits en los enrutadores, por lo que se quedaron sin uso durante muchos años. Cuando se diseñaron los servicios diferenciados, la IETF tiró la toalla y reutilizó este campo. Ahora, los 6 bits superiores se utilizan para marcar el paquete con su clase de servicio; anteriormente en este capítulo describimos los servicios expedito y asegurado. Los 2 bits inferiores se utilizan para transportar información sobre la notificación de congestión; por ejemplo, si el paquete ha experimentado congestión o no. Asimismo, también describimos la notificación explícita de congestión como parte del control de la congestión.

El campo *Longitud total* incluye todo en el datagrama: tanto el encabezado como los datos. La longitud máxima es de 65 535 bytes. En la actualidad este límite es tolerable, pero con las redes futuras tal vez se requieran datagramas más grandes.

El campo *Identificación* es necesario para que el host de destino determine a qué paquete pertenece un fragmento recién llegado. Todos los fragmentos de un paquete contienen el mismo valor de *Identificación*.

A continuación viene un bit sin uso, lo cual es sorprendente, ya que el espacio en el encabezado IP es muy escaso. Como broma del día de los inocentes, Bellovin (2003) propuso usar este bit para detectar el tráfico malicioso. Esto simplificaría de manera considerable la seguridad, puesto que se sabría que los paquetes con el bit "malo" habían sido enviados por atacantes y sólo había que descartarlos. Por desgracia, la seguridad de las redes no es tan simple.

Después vienen dos campos de 1 bit relacionados con la fragmentación. *DF* significa no fragmentar (*Don't Fragment*), es una orden para que los enrutadores no fragmenten el paquete. En un principio estaban destinados para soportar a los hosts incapaces de reunir las piezas otra vez. Ahora se utiliza como parte del proceso para descubrir la MTU de la ruta: el paquete más grande que puede viajar a través de una ruta sin necesidad de fragmentarse. Al marcar el datagrama con el bit *DF*, el emisor sabe que llegará en una pieza o recibirá de vuelta un mensaje de error.

MF significa más fragmentos (*More Fragments*). Todos los fragmentos excepto el último tienen establecido este bit, que es necesario para saber cuándo han llegado todos los fragmentos de un datagrama.

El *Desplazamiento del fragmento* indica a qué parte del paquete actual pertenece este fragmento. Todos los fragmentos excepto el último del datagrama deben ser un múltiplo de 8 bytes, que es la unidad de fragmentos elemental. Dado que se proporcionan 13 bits, puede haber un máximo de 8 192 fragmentos

por datagrama, para soportar una longitud máxima de paquete de hasta el límite del campo *Longitud total*. En conjunto, los campos *Identificación*, *MF* y *Desplazamiento del fragmento* se utilizan para implementar la fragmentación según se describe en la sección 5.5.5.

El campo *TtL* (*Tiempo de vida*) es un contador que se utiliza para limitar el tiempo de vida de un paquete. En un principio se suponía que iba a contar el tiempo en segundos, lo cual permitía un periodo de vida máximo de 255 seg. Hay que decrementarlo en cada salto y se supone que se decrementa muchas veces cuando un paquete se pone en cola durante un largo tiempo en un enrutador. En la práctica, simplemente cuenta los saltos. Cuando el contador llega a cero, el paquete se descarta y se envía de regreso un paquete de aviso al host de origen. Esta característica evita que los paquetes anden vagando eternamente, algo que de otra manera podría ocurrir si se llegaran a corromper las tablas de enrutamiento.

Una vez que la capa de red ha ensamblado un paquete completo, necesita saber qué hacer con él. El campo *Protocolo* le indica a cuál proceso de transporte debe entregar el paquete. TCP es una posibilidad, pero también están UDP y otros más. La numeración de los protocolos es global en toda la Internet. Anteriormente los protocolos y otros números asignados se listaban en el RFC 1700, pero en la actualidad están contenidos en una base de datos en línea localizada en www.iana.org.

Puesto que el encabezado transporta información vital, como las direcciones, estima su propia suma de verificación por protección, la *Suma de verificación del encabezado*. El algoritmo suma todas las medias palabras de 16 bits del encabezado a medida que vayan llegando, mediante el uso de la aritmética de complemento a uno, y después obtiene el complemento a uno del resultado. Para los fines de este algoritmo, se supone que la *Suma de verificación del encabezado* es cero al momento de la llegada. Dicha suma de verificación es útil para detectar errores mientras el paquete viaja por la red. Tenga en cuenta que se debe recalcular en cada salto, ya que por lo menos hay un campo que siempre cambia (el campo *Tiempo de vida*), aunque se pueden usar trucos para agilizar ese cálculo.

Los campos *Dirección de origen* y *Dirección de destino* indican la dirección IP de las interfaces de red de la fuente y del destino. En la siguiente sección estudiaremos las direcciones de Internet.

El campo *Opciones* se diseñó para proporcionar un recurso que permitiera que las versiones subsiguientes del protocolo incluyeran información que no estuviera presente en el diseño original, para que los experimentadores puedan probar ideas nuevas y evitar la asignación de bits de encabezado a la información que se necesite muy poco. Las opciones son de longitud variable. Cada una empieza con un código de 1 byte que identifica la opción. Algunas opciones van seguidas de un campo de longitud de la opción de 1 byte, y luego de uno o más bytes de datos. El campo *Opciones* se rellena para completar múltiplos de 4 bytes. En un principio se definieron las cinco opciones que se listan en la figura 5-47.

La opción *Seguridad* indica qué tan secreta es la información. En teoría, un enrutador militar podría usar este campo para especificar que no se enruten paquetes a través de ciertos países que los militares

Opción	Descripción
Seguridad.	Especifica qué tan secreto es el datagrama.
Enrutamiento estricto desde el origen.	Proporciona la ruta completa a seguir.
Enrutamiento libre desde el origen.	Proporciona una lista de enrutadores que no se deben omitir.
Registrar ruta.	Hace que cada enrutador adjunte su dirección IP.
Estampa de tiempo.	Hace que cada enrutador adjunte su dirección y su etiqueta de tiempo.

Figura 5-47. Algunas de las opciones del protocolo IP.

consideren “malos”. En la práctica todos los enrutadores lo ignoran, por lo que su única función real es la de ayudar a los espías a encontrar la información importante con mayor facilidad.

La opción de *Enrutamiento estricto desde el origen* proporciona la ruta completa desde el origen hasta el destino como una secuencia de direcciones IP. Se requiere que el datagrama siga esa ruta exacta. Esta opción se usa sobre todo cuando los administradores de sistemas necesitan enviar paquetes de emergencia cuando se corrompen las tablas de enrutamiento, o para hacer mediciones de sincronización.

La opción de *Enrutamiento libre desde el origen* requiere que el paquete pase por los enrutadores indicados en la lista, y en el orden especificado, pero se le permite pasar a través de otros enrutadores en el camino. Por lo general esta opción sólo indicará unos cuantos enrutadores para forzar una ruta específica. Por ejemplo, si se desea obligar a un paquete de Londres a Sydney a ir hacia el Oeste en lugar de hacia el Este, esta opción podría especificar enrutadores en Nueva York, Los Ángeles y Honolulú. Esta opción es de mucha utilidad cuando las consideraciones políticas o económicas obligan a pasar a través de, o evitar, ciertos países.

La opción de *Registrar ruta* indica a cada enrutador a lo largo de la ruta que adjunte su dirección IP al campo *Opciones*. Esto permite a los administradores del sistema buscar fallas en los algoritmos de enrutamiento (“¿Por qué todos los paquetes de Houston a Dallas pasan primero por Tokio?”). Cuando se estableció ARPANET en un principio, ningún paquete pasaba nunca por más de nueve enrutadores, por lo que 40 bytes de opciones eran más que suficientes. Pero como dijimos antes, ahora son muy pocos.

Por último, la opción *Estampa de tiempo* es como la opción *Registrar ruta*, excepto que además de registrar su dirección IP de 32 bits, cada enrutador también registra una estampa de tiempo de 32 bits. Esta opción también es principalmente útil para la medición en las redes.

Hoy en día se han dejado de usar las opciones IP. La mayoría de los enrutadores las ignoran o no las procesan en forma eficiente, haciéndolas a un lado como un caso poco común. Es decir, sólo cuentan con soporte parcial y se usan muy raras veces.

5.6.2 Direcciones IP

Una característica que define a IPv4 consiste en sus direcciones de 32 bits. Cada host y enrutador de Internet tiene una dirección IP que se puede usar en los campos *Dirección de origen* y *Dirección de destino* de los paquetes IP. Es importante tener en cuenta que una dirección IP en realidad no se refiere a un host, sino a una interfaz de red, por lo que si un host está en dos redes, debe tener dos direcciones IP. Sin embargo, en la práctica la mayoría de los hosts están en una red y, por ende, tienen una dirección IP. En contraste, los enrutadores tienen varias interfaces y, por lo tanto, múltiples direcciones IP.

Prefijos

A diferencia de las direcciones Ethernet, las direcciones IP son jerárquicas. Cada dirección de 32 bits está compuesta de una porción de red de longitud variable en los bits superiores, y de una porción de host en los bits inferiores. La porción de red tiene el mismo valor para todos los hosts en una sola red, como una LAN Ethernet. Esto significa que una red corresponde a un bloque contiguo de espacio de direcciones IP. A este bloque se le llama **prefijo**.

Las direcciones IP se escriben en **notación decimal con puntos**. En este formato, cada uno de los 4 bytes se escribe en decimal, de 0 a 255. Por ejemplo, la dirección hexadecimal 80D00297 de 32 bits se escribe como 128.208.2.151. Para escribir los prefijos, se proporciona la dirección IP menor en el bloque y el tamaño del mismo. El tamaño se determina mediante el número de bits en la porción de red; el resto de los bits en la porción del host pueden variar. Esto significa que el tamaño debe ser una potencia de dos.

Por convención, el prefijo se escribe después de la dirección IP como una barra diagonal seguida de la longitud en bits de la porción de red. En nuestro ejemplo, si el prefijo contiene 2^8 direccciones y, por lo tanto, deja 24 bits para la porción de red, se escribe como 128.208.0.0/24.

Como la longitud del prefijo no se puede inferir sólo a partir de la dirección IP, los protocolos de enrutamiento deben transportar los prefijos hasta los enrutadores. Algunas veces los prefijos se describen simplemente mediante su longitud, como en un "/16" que se pronuncia como "*slash* 16". La longitud del prefijo corresponde a una máscara binaria de 1s en la porción de red. Cuando se escribe de esta forma, se denomina **máscara de subred**. Se puede aplicar un AND a la máscara de subred con la dirección IP para extraer sólo la porción de la red. Para nuestro ejemplo, la máscara de subred es 255.255.255.0. La figura 5-48 muestra un prefijo y una máscara de subred.

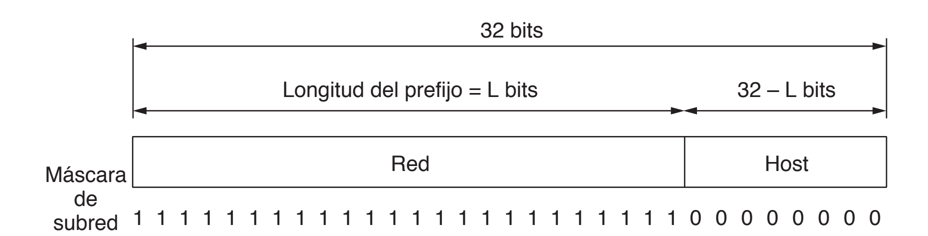


Figura 5-48. Un prefijo y una máscara de subred del protocolo IP.

Las direcciones jerárquicas tienen ventajas y desventajas considerables. La ventaja clave de los prefijos es que los enrutadores pueden reenviar paquetes con en base en la porción de red de la dirección, siempre y cuando cada una de las redes tenga un bloque de direcciones único. La porción del host no importa a los enrutadores, ya que enviarán en la misma dirección a todos los hosts de la misma red. Sólo hasta que los paquetes llegan a la red de destino es cuando se reenvían al host correcto. Esto hace que las tablas de enrutamiento sean mucho más pequeñas de lo que podrían ser con cualquier otro método. Considere que el número de hosts en Internet se está aproximando a los 1000 millones. Ésta sería una tabla muy grande como para que todos los enrutadores tuvieran que guardarla. Sin embargo, mediante el uso de una jerarquía los enrutadores sólo necesitan mantener las rutas para cerca de 300 000 prefijos.

Aunque el uso de una jerarquía permite a Internet escalar, tiene dos desventajas. En primer lugar, la dirección IP de un host depende de su ubicación en la red. Una dirección Ethernet se puede usar en cualquier parte del mundo, pero cada dirección IP pertenece a una red específica, por lo que los enrutadores sólo podrán entregar paquetes destinados a esa dirección en la red. Se necesitan diseños como IP móvil para soportar hosts que se desplacen de una red a otra, pero que deseen mantener las mismas direcciones IP.

La segunda desventaja es que la jerarquía desperdicia direcciones a menos que se administre con cuidado. Si se asignan direcciones a las redes en bloques (muy) grandes, habrá (muchas) direcciones que se asignen pero no se utilicen. Esta asignación no sería de gran importancia si hubiera muchas direcciones para repartir. Sin embargo, hace más de dos décadas se descubrió que el tremendo crecimiento de Internet estaba agotando con rapidez el espacio de direcciones libres. IPv6 es la solución a este agotamiento, pero hasta que no se implemente de manera amplia habrá mucha presión por asignar direcciones IP, de manera que se utilicen con mucha eficiencia.

Subredes

Para evitar conflictos, los números de red se administran a través de una corporación sin fines de lucro llamada **ICANN** (**Corporación de Internet para la Asignación de Nombres y Números**, del inglés

Internet Corporation for Assigned Names and Numbers). Esta corporación ha delegado partes de este espacio de direcciones a varias autoridades regionales, las cuales reparten las direcciones IP a los ISP y otras compañías. Éste es el proceso por el cual se asigna un bloque de direcciones IP a una compañía.

Sin embargo, este proceso es sólo el principio de la historia, puesto que la asignación de direcciones IP es continua a medida que crecen las compañías. Ya dijimos antes que el enrutamiento por prefijo requiere que todos los hosts en una red tengan el mismo número de red. Esta propiedad puede provocar problemas a medida que las redes aumentan su tamaño. Por ejemplo, considere una universidad que empezó con nuestro prefijo de ejemplo /16, para que el Departamento de Ciencias Computacionales lo use en las computadoras de su red Ethernet. Un año después, el Departamento de Ingeniería eléctrica desea entrar a Internet. Pronto le sigue el Departamento de Arte. ¿Qué direcciones IP deben usar estos departamentos? Para obtener más bloques hay que ir fuera de la universidad y puede ser costoso o inconveniente. Además, el prefijo /16 que ya está asignado tiene suficientes direcciones para más de 60 000 hosts. Tal vez se tenga pensado un crecimiento considerable, pero mientras esto no ocurra sería un desperdicio asignar más bloques de direcciones IP a la misma universidad. Se requiere una organización diferente.

La solución es dividir el bloque de direcciones en varias partes para uso interno en forma de múltiples redes, pero actuar como una sola red para el mundo exterior. A estas partes de la red se les llama **subredes**. Como mencionamos en el capítulo 1, hay que estar conscientes de que este nuevo uso del término entra en conflicto con el uso anterior de "subred", cuyo significado es el conjunto de todos los enrutadores y líneas de comunicación en una red.

La figura 5-49 muestra cómo pueden ayudar las subredes en nuestro ejemplo. El prefijo /16 se dividió en varias piezas. Esta división no necesita ser uniforme, pero hay que alinear cada pieza de manera que se pueda usar cualquier bit en la porción inferior del host. En este caso, la mitad del bloque (un /17) se asigna al Departamento de Ciencias Computacionales, una cuarta parte se asigna al Departamento de Ingeniería Eléctrica (un /18) y una octava parte (un /19) al Departamento de Arte. El octavo restante queda sin asignar. Una manera distinta de ver cómo se dividió el bloque es analizar los prefijos resultantes cuando se escriben en notación binaria:

Ciencias Computacionales:	10000000	11010000	1|xxxxxxx	xxxxxxxx
Ingeniería Eléctrica:	10000000	11010000	00|xxxxxx	xxxxxxxx
Arte:	10000000	11010000	011|xxxxx	xxxxxxxx

Aquí, la barra vertical (|) muestra el límite entre el número de la subred y la porción del host.

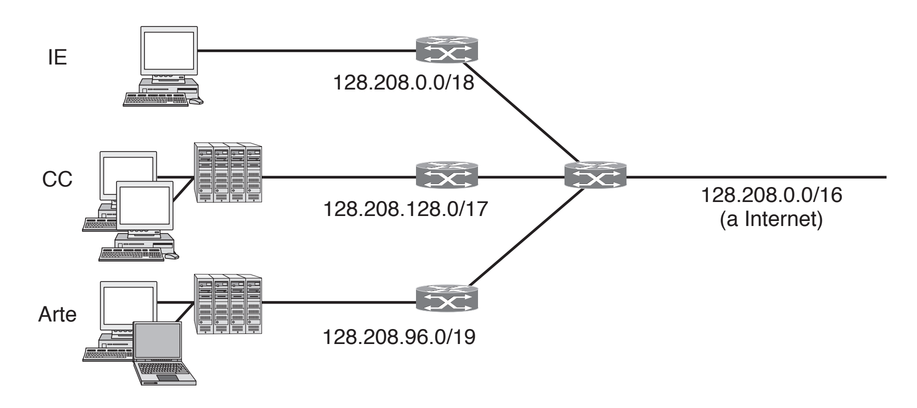


Figura 5-49. División de un prefijo IP en redes separadas mediante el uso de subredes.

Cuando llega un paquete al enrutador principal, ¿cómo sabe a cuál subred entregarlo? Aquí es donde entran los detalles de nuestros prefijos. Una forma sería que cada enrutador tuviera una tabla con 65 535 entradas que le indicaran qué línea de salida usar para cada host en el campus. Pero esto socavaría el principal beneficio de escalamiento que obtenemos al usar una jerarquía. En cambio, los enrutadores simplemente tienen que conocer las máscaras de subred para las redes en el campus.

Cuando llega un paquete, el enrutador analiza su dirección de destino y verifica a qué subred pertenece. Para ello, el enrutador aplica un AND a la dirección del destino con la máscara para cada subred y verifica que el resultado sea el prefijo correspondiente. Por ejemplo, considere un paquete destinado para la dirección IP 128.208.2.151. Para ver si está dirigido al Departamento de Ciencias Computacionales, le aplicamos un AND con 255.255.128.0 para tomar los primeros 17 bits (que son 128.208.0.0) y ver si coinciden con la dirección del prefijo (que es 128.208.128.0). En este caso no coinciden. Si verificamos los primeros 18 bits para el Departamento de Ingeniería Eléctrica, obtenemos 128.208.0.0 al aplicar un AND entre éstos y la máscara de subred. Este resultado coincide con la dirección del prefijo, por lo que el paquete se reenvía por la interfaz que conduce a la red de ingeniería eléctrica.

Las divisiones de las subredes se pueden cambiar más adelante si es necesario, para lo cual se actualizan todas las máscaras de subred en los enrutadores dentro de la universidad. Fuera de la red, las subredes no son visibles, por lo que para asignar una nueva subred no hay que ponerse en contacto con la ICANN ni cambiar bases de datos externas.

CIDR: Enrutamiento Interdominio sin Clases

Incluso si se asignan bloques de direcciones IP de manera que las direcciones se utilicen con eficiencia, de todas formas hay un problema presente: la explosión de las tablas de enrutamiento.

Los enrutadores en las organizaciones en el extremo de una red, como una universidad, necesitan tener una entrada para cada una de sus subredes, de manera que el enrutador pueda saber qué línea usar para llegar a esa red. Para las rutas a destinos fuera de la organización, sólo necesitan usar la regla simple predeterminada de enviar los paquetes en la línea hacia el ISP que conecta a la organización con el resto de Internet. Las otras direcciones de destino deben estar por ahí en alguna parte.

Los enrutadores en los ISP y las redes troncales en medio de Internet no se pueden dar esos lujos. Ellos deben saber qué ruta tomar hacia cada una de las redes; aquí no funciona la regla simple predeterminada. Se dice que estos enrutadores básicos están en la **zona libre predeterminada** de Internet. Nadie sabe en realidad cuántas redes están conectadas a Internet, pero es una gran cantidad, por lo menos un millón. Esto puede generar una tabla muy grande. Tal vez no parezca muy grande según los estándares computacionales, pero hay que tener en cuenta que los enrutadores deben realizar una búsqueda en esta tabla para reenviar cada paquete, y los enrutadores en los ISP grandes pueden reenviar hasta millones de paquetes por segundo. Se requiere un hardware especializado y una memoria rápida para procesar paquetes a estas tasas de transmisión, no una computadora de propósito general.

Además, los algoritmos de enrutamiento requieren que cada enrutador intercambie información sobre las direcciones a las que puede llegar con otros enrutadores. Entre más grandes sean las tablas, más información habrá que comunicar y procesar. El procesamiento aumenta por lo menos en forma lineal con respecto al tamaño de la tabla. Una mayor comunicación aumenta la probabilidad de que algunas partes se pierdan, por lo menos en forma temporal, lo que tal vez conduzca a inestabilidades en el enrutamiento.

El problema de las tablas de enrutamiento se podría haber resuelto mediante el uso de una jerarquía más profunda, como la red telefónica. Por ejemplo, hacer que cada dirección IP contenga campos de país, estado/provincia, ciudad, red y host podría funcionar. Así, cada enrutador sólo tendría que saber cómo llegar a cada país, a los estados o provincias en su propio país, a las ciudades en su estado o provincia, y a las redes en su ciudad. Por desgracia, esta solución requeriría mucho más de 32 bits para las direcciones

IP y utilizaría las direcciones de una manera ineficiente (además, Liechtenstein tendría tantos bits en sus direcciones como Estados Unidos).

Por fortuna, hay algo que podemos hacer para reducir los tamaños de las tablas de enrutamiento. Podemos aplicar la misma perspectiva que en las subredes: los enrutadores en distintas ubicaciones pueden saber acerca de una dirección IP dada que pertenece a prefijos de distintas zonas. Sin embargo, en vez de dividir un bloque de direcciones en subredes, aquí combinamos varios prefijos pequeños en un solo prefijo más grande. A este proceso se le conoce como **agregación de rutas**. Algunas veces al prefijo más grande resultante se le denomina **superred** para contrastar con las subredes como la división de bloques de direcciones.

Con la agregación, las direcciones IP están contenidas en prefijos de diversos tamaños. La misma dirección IP que un enrutador trata como parte de un prefijo /22 (un bloque que contiene 2^{10} direcciones), puede ser tratada por otro enrutador como parte de un prefijo /20 más grande (que contiene 2^{12} direcciones). Es responsabilidad de cada enrutador tener la información del prefijo correspondiente. Este diseño funciona con las subredes y se conoce como **CIDR** (**Enrutamiento Interdominio sin Clases**, del inglés *Classless InterDomain Routing*). La versión más reciente se especifica en el RFC 4632 (Fuller y Li, 2006). El nombre resalta el contraste con las direcciones que codifican la jerarquía con las clases, lo cual describiremos en breve.

Para facilitar el entendimiento de CIDR, vamos a considerar un ejemplo en el que hay un bloque de 8 192 direcciones IP disponible, empezando en 194.24.0.0. Suponga que la Universidad de Cambridge necesita 2 048 direcciones y se le asignan las direcciones 194.24.0.0 a 194.24.7.255, junto con la máscara 255.255.248.0. Éste es un prefijo /21. A continuación, la Universidad de Oxford pide 4 096 direcciones. Como un bloque de 4 096 direcciones debe estar en un límite de 4 096 bytes, a Oxford no se le pueden asignar direcciones que empiecen en 192.24.8.0. Sin embargo, recibe las direcciones 192.24.16.0 a 192.24.31.255, junto con la máscara de subred 255.255.240.0. Por último, la Universidad de Edimburgo pide 1024 direcciones y se le asignan las direcciones 192.24.8.0 a 194.24.11.255 junto con la máscara 255.255.252.0. En la figura 5-50 se sintetizan estas asignaciones.

Universidad	Primera dirección	Última dirección	Cuántas	Prefijo
Cambridge	194.24.0.0	194.24.7.255	2048	194.24.0.0/21
Edimburgo	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
(Disponible)	194.24.12.0	194.24.15.255	1024	194.24.12.0/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/20

Figura 5-50. Un conjunto de asignaciones de direcciones IP.

A todos los enrutadores en la zona libre predeterminada se les dice ahora sobre las direcciones IP en las tres redes. Los enrutadores cercanos a las universidades tal vez necesiten enviar por una línea de salida distinta para cada uno de los prefijos, así que ellos necesitan una entrada para cada uno de de los prefijos en sus tablas de enrutamiento. Un ejemplo es el enrutador en Londres de la figura 5-51.

Ahora veamos estas tres universidades desde el punto de vista de un enrutador distante en Nueva York. Todas las direcciones IP en los tres prefijos se deben enviar de Nueva York (o de Estados Unidos en general) a Londres. El proceso de enrutamiento en Londres detecta esto y combina los tres prefijos en una sola entrada agregada para el prefijo 194.24.0.0/19, que pasa al enrutador de Nueva York. Este prefijo contiene 8K direcciones y cubre las tres universidades, junto con las otras 1024 direcciones no asignadas. Mediante el uso de la agregación, estos prefijos se redujeron a uno, con lo cual se redujeron tanto los

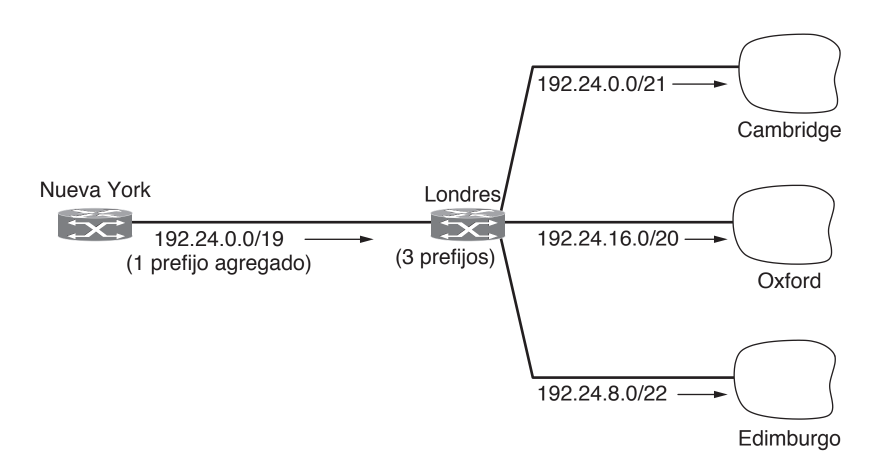


Figura 5-51. Agregación de prefijos IP.

prefijos que se deben informar al enrutador de Nueva York, como las entradas en la tabla de enrutamiento de éste.

Una vez que se activa la agregación, es un proceso automático que depende de qué prefijos están ubicados en qué parte de Internet, no en las acciones de un administrador que asigna direcciones a las redes. La agregación se utiliza mucho en Internet y puede reducir el tamaño de las tablas de los enrutadores a cerca de 200 000 prefijos.

Como una sorpresa adicional, es posible traslapar a los prefijos. La regla es que los paquetes se envíen en la dirección de la ruta más específica, o el **prefijo más largo coincidente** que tenga la menor cantidad de direcciones IP. El enrutamiento del prefijo más largo coincidente ofrece un grado conveniente de flexibilidad, como podemos ver en el comportamiento del enrutador de Nueva York en la figura 5-52. Este enrutador aún utiliza un solo prefijo agregado para enviar tráfico de las tres universidades a Londres. Sin embargo, el bloque previamente disponible de direcciones dentro de este prefijo se ha asignado ahora a una red en San Francisco. Una posibilidad es que el enrutador de Nueva York mantenga cuatro prefijos, y que envíe los paquetes de tres de ellos a Londres y los paquetes del cuarto prefijo a San Francisco. Sin embargo, el enrutamiento del prefijo más largo coincidente puede manejar este reenvío de paquetes con los dos prefijos que se muestran. Un prefijo general se utiliza para dirigir el tráfico de todo el bloque a Londres. También se utiliza uno más específico para dirigir una parte del prefijo más grande a San Francisco. Con la regla del prefijo más largo coincidente, las direcciones IP dentro de la red de San Francisco se enviarán en la línea de salida a San Francisco, y todas las demás direcciones IP en el prefijo más grande se enviarán a Londres.

En concepto, el CIDR funciona de la siguiente manera. Cuando llega un paquete, se explora la tabla de enrutamiento para determinar si el destino está dentro del prefijo. Es posible que coincidan varias

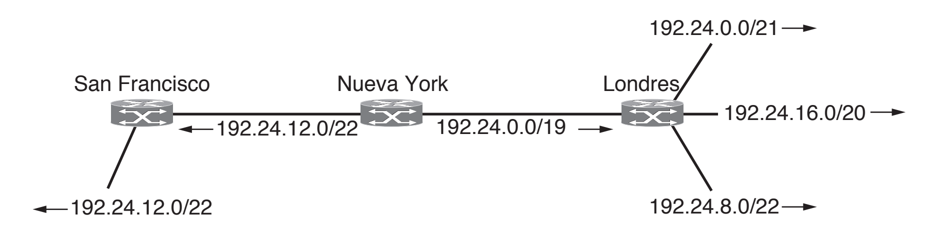


Figura 5-52. Enrutamiento del prefijo más largo coincidente en el enrutador de Nueva York.

entradas con distintas longitudes de prefijos, en cuyo caso se utiliza la entrada con el prefijo más largo. Por ende, si hay una coincidencia para una máscara /20 y una máscara /24, se utiliza la entrada /24 para buscar la línea de salida para el paquete. Sin embargo, este proceso sería tedioso si la tabla realmente se explorara entrada por entrada. En cambio, se han ideado algoritmos complejos para agilizar el proceso de coincidencia de direcciones (Ruiz-Sánchez y colaboradores, 2001). Los enrutadores comerciales usan chips VLSI personalizados con estos algoritmos incrustados en el hardware.

Direccionamiento con clases y especial

Para ayudar al lector a que aprecie mejor la razón de por qué CIDR es tan útil, relataremos brevemente el diseño anterior a éste. Antes de 1993, las direcciones IP se dividían en las cinco categorías listadas en la figura 5-53. Esta asignación se denominó **direccionamiento con clases**.

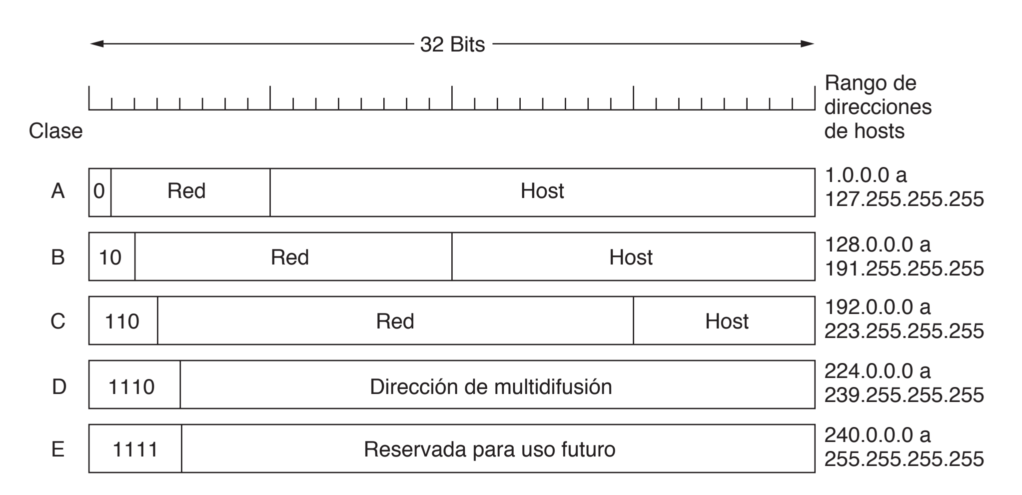


Figura 5-53. Formatos de direcciones IP.

Los formatos de clase A, B y C permiten hasta 128 redes con 16 millones de hosts cada una, 16 384 redes con hasta 65 536 hosts cada una y 2 millones de redes (por ejemplo, LAN) con hasta 256 hosts cada una (aunque unas cuantas de éstas son especiales). También se soporta la multidifusión (el formato de clase D), en el cual un datagrama se dirige a múltiples hosts. Las direcciones que empiezan con 1111 se reservan para un uso futuro. Serían valiosas para usar ahora, dado que se está agotando el espacio de direcciones IPv4. Por desgracia, muchos hosts no aceptarán estas direcciones como válidas, puesto que han estado fuera de los límites por tanto tiempo y es difícil enseñar nuevos trucos a los hosts viejos.

Éste es un diseño jerárquico, pero a diferencia de CIDR, los tamaños de los bloques de direcciones son fijos. Existen más de 2 000 millones de direcciones, pero al organizar el espacio de direcciones en clases se desperdician millones de ellas. En particular, el verdadero villano es la red clase B. Para la mayoría de las organizaciones, una red clase A con 16 millones de direcciones es demasiado grande, y una red clase C con 256 direcciones es muy pequeña. Una red clase B con 65 536 direcciones es justo lo necesario. En el folclor de Internet, esta situación se conoce como el **problema de los tres osos** [del cuento *Ricitos de oro y los tres osos* (Southey, 1848)].

En realidad, una dirección clase B es demasiado grande para la mayoría de las organizaciones. Hay estudios que demuestran que más de la mitad de todas las redes clase B tienen menos de 50 hosts. Una red clase C habría bastado, pero sin duda todas las organizaciones que solicitaron una dirección clase B

pensaron que un día el campo de hosts de 8 bits les quedaría pequeño. En retrospectiva, podría haber sido mejor que las redes clase C usaran 10 bits en lugar de 8 para el número de host, permitiendo 1022 hosts por red. De haber sido éste el caso, la mayoría de las organizaciones probablemente se habría conformado con una red clase C y hubiera existido medio millón de ellas (en vez de sólo 16 384 redes clase B).

Es duro culpar a los diseñadores de Internet por no haber proporcionado más (y más pequeñas) direcciones de clase B. En el momento en que se tomó la decisión de crear las tres clases, Internet era una red de investigación que conectaba las principales universidades de investigación en Estados Unidos (más un número muy pequeño de compañías y sitios del ejército que hacían investigación de redes). En esa época nadie percibía que Internet llegaría a ser un sistema de comunicación de mercado masivo que rivalizaría con la red telefónica. También en esa época alguien dijo sin dudar: "Estados Unidos tiene alrededor de 2 000 universidades. Aun cuando todas se conectaran a Internet y muchas universidades en otros países también, nunca llegaremos a 16 000 puesto que no hay tantas universidades en todo el mundo. Además, como el número de host es un número integral de bytes, agiliza el procesamiento de los paquetes" (lo que entonces se hacía completamente mediante software). Tal vez algún día las personas vean hacia atrás y culpen a los que diseñaron el esquema de números telefónicos diciendo: "Qué idiotas, ¿por qué no incluyeron el número de planeta en el número telefónico?" Pero en ese tiempo, no parecía necesario.

Para lidiar con estos problemas se introdujeron las subredes para asignar de manera flexible bloques de direcciones dentro de una organización. Después se agregó el CIDR para reducir el tamaño de la tabla de enrutamiento global. Hoy en día, los bits que indican si una dirección IP pertenece a una red clase A, B o C ya no se utilizan, aunque las referencias a estas clases en la literatura siguen siendo comunes.

Para ver cómo el hecho de descartar las clases complicó aún más el reenvío de paquetes, considere lo simple que era en el viejo sistema con clases. Cuando llegaba un paquete en un enrutador, se desplazaba una copia de la dirección IP 28 bits a la derecha para producir un número de clase de 4 bits. Después una ramificación de 16 vías ordenaba los paquetes en las clases A, B, C (junto con D y E), con ocho casos para la clase A, cuatro casos para la clase B y dos casos para la clase C. Después se enmascaraba el código para cada clase del número de red de 8, 16 o 24 bits y se alineaba a la derecha en una palabra de 32 bits. Luego se buscaba el número de red en la tabla A, B o C, por lo general mediante el indexado para las redes A y B, y el *hashing* para las redes C. Una vez que se encontraba la entrada, se podía buscar la línea de salida y reenviar el paquete. Esto es mucho más simple que la operación del prefijo más largo coincidente, el cual ya no puede usar una simple búsqueda en una tabla debido a que una dirección IP puede tener un prefijo de cualquier longitud.

Las direcciones de clase D se siguen utilizando en Internet para la multidifusión. En realidad podría ser más preciso decir que se están empezando a usar para multidifusión, puesto que en el pasado la multidifusión en Internet no se ha implementado mucho.

También existen otras direcciones con significados especiales, como se muestra en la figura 5-54. La dirección IP 0.0.0.0, que es la dirección más baja, es utilizada por los hosts al momento de encenderlos. Significa "esta red" o "este host". Las direcciones IP con 0 como número de red se refieren a la red actual. Estas direcciones permiten que las máquinas hagan referencia a su propia red sin conocer su número (pero tienen que conocer la máscara de red para saber cuántos 0s incluir). La dirección que consiste sólo en 1s, o 255.255.255.255 (la dirección más alta), se utiliza para indicar a todos los hosts en la red especificada. Permite la difusión en la red local, por lo general una LAN. Las direcciones con un número de red apropiado y 1s en el campo de host permiten a las máquinas enviar paquetes de difusión a redes LAN distantes en cualquier parte de Internet. Sin embargo, muchos administradores de red deshabilitan esta característica, ya que es sobre todo un peligro de seguridad. Por último, todas las direcciones de la forma 127.*xx.yy.zz* se reservan para las pruebas de *loopback*. Los paquetes que se envían a esa dirección no se ponen en el cable; se procesan en forma local y se tratan como si fueran paquetes entrantes. Esto permite enviar paquetes al host sin que el emisor conozca su número, lo cual es útil para realizar pruebas.

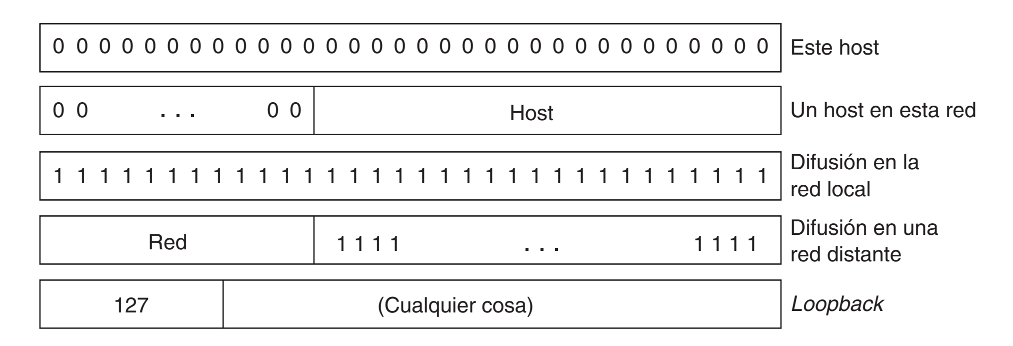


Figura 5-54. Direcciones IP especiales.

NAT: Traducción de Dirección de Red

Las direcciones IP son escasas. Un ISP podría tener una dirección con prefijo de /16, lo cual le da 65 534 números de host. Si tiene más clientes que esos, tiene un problema.

Esta escasez ha conducido a técnicas para usar las direcciones IP con moderación. Un método es asignar dinámicamente una dirección IP a una computadora cuando ésta se encuentra encendida y usa la red, y tomar de vuelta la dirección IP cuando el host se vuelve inactivo. Así, la dirección IP se puede asignar a otra computadora que se active en ese momento. De esta manera, una sola dirección de /16 puede manejar hasta 65 534 usuarios activos.

Esta estrategia funciona bien en algunos casos, por ejemplo, para las redes de marcación, las computadoras móviles y otras computadoras que pueden estar temporalmente ausentes o apagadas. Sin embargo, no funciona muy bien para los clientes comerciales. Muchas PC en negocios deben estar prendidas en forma continua. Algunas son máquinas de los empleados, que se respaldan en la noche, y otras son servidores que tal vez necesiten atender una solicitud remota sin previo aviso. Estos negocios tienen una línea de acceso que siempre proporciona conectividad al resto de Internet.

Esta situación se aplica cada vez más a los usuarios domésticos que se suscriben a ADSL o Internet por cable, ya que no hay un cargo por conexión (sólo una tarifa mensual fija). Muchos de estos usuarios tienen dos o más computadoras en su hogar, a menudo una para cada miembro de la familia, y todos quieren estar en línea todo el tiempo. La solución es conectar todas las computadoras en una red doméstica a través de una LAN y colocar un enrutador (inalámbrico) en ella. Así, el enrutador se conecta con el ISP. Desde el punto de vista del ISP, la familia es ahora lo mismo que un pequeño negocio con un puñado de computadoras. “Bienvenido a González, S.A”. Con las técnicas que hemos visto hasta ahora, cada computadora debe tener su propia dirección IP todo el tiempo. Para un ISP con muchos miles de clientes, en especial clientes comerciales y familias que son casi como pequeños negocios, la demanda de direcciones IP puede exceder rápidamente el bloque disponible.

El problema de quedarse sin direcciones IP no es uno teórico que podría ocurrir en cierto momento en un futuro distante. Está ocurriendo justo aquí y ahora. La solución a largo plazo es que toda la Internet migre a IPv6, que cuenta con direcciones de 128 bits. Esta transición está ocurriendo lentamente, pero pasarán años antes de que el proceso esté completo. Mientras tanto, para sobrevivir se requería una solución rápida. Esta solución, que se utiliza ampliamente en la actualidad, se conoce como **NAT** (**Traducción de Dirección de Red**, del inglés *Network Address Translation*) y se describe en el RFC 3022; el cual se resume a continuación. Para obtener información adicional, consulte a Dutcher (2001).

La idea básica detrás de NAT es que el ISP asigne a cada hogar o negocio una sola dirección IP (o a lo más, una pequeña cantidad de éstas) para el tráfico de Internet. *Dentro* de la red del cliente, cada computadora obtiene una dirección IP única, la cual se utiliza para enrutar el tráfico interno. Sin embargo,

justo antes de que un paquete salga de la red del cliente y vaya al ISP, la dirección IP única interna se traduce a la dirección IP pública compartida. Esta traducción hace uso de los tres rangos de direcciones IP que se han declarado como privados. Las redes pueden utilizarlos de manera interna como deseen. La única regla es que no pueden aparecer paquetes que contengan estas mismas direcciones en Internet. Los tres rangos reservados son:

10.0.0.0	– 10.255.255.255/8	(16,777,216 hosts)
172.16.0.0	– 172.31.255.255/12	(1,048,576 hosts)
192.168.0.0	– 192.168.255.255/16	(65,536 hosts)

El primer rango contiene 16 777 216 direcciones (excepto por las que contienen sólo 1s o 0s, como siempre) y es la opción común, incluso aunque la red no sea tan grande.

En la figura 5-55 se muestra la operación de NAT. Dentro de las premisas del cliente, cada máquina tiene una dirección única de la forma 10.*x*.*y*.*z*. Sin embargo, antes de que un paquete salga de las premisas del cliente, pasa a través de una **caja NAT** que convierte la dirección IP de origen interna, 10.0.0.1 en la figura, a la dirección IP verdadera del cliente, 198.60.42.12 en este ejemplo. A menudo, la caja NAT se combina en un solo dispositivo con un *firewall* (corta fuegos) que proporciona seguridad controlando cuidadosamente lo que entra y sale por la red de la compañía. En el capítulo 8 estudiaremos los *firewalls*. También es posible integrar la caja NAT en un enrutador o módem ADSL.

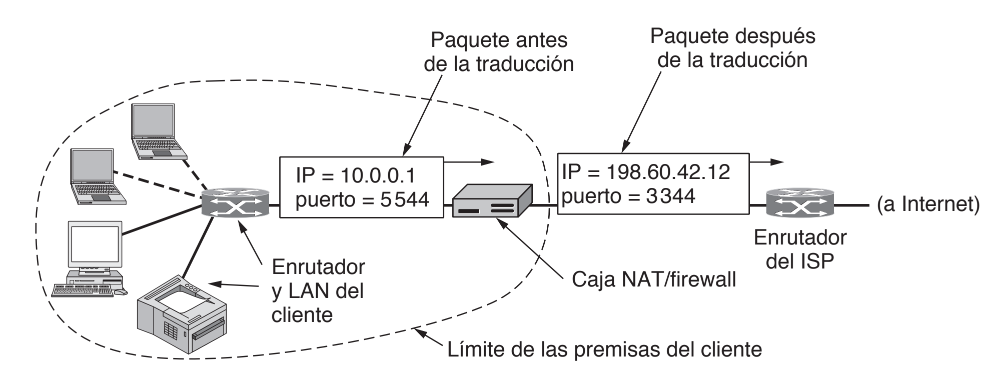


Figura 5-55. Colocación y funcionamiento de una caja NAT.

Hasta ahora hemos ignorado un detalle pequeño pero crucial: cuando la respuesta vuelve (por ejemplo, de un servidor web), se dirige naturalmente a 198.60.42.12, por lo que, ¿cómo sabe ahora la caja NAT con qué dirección interna se reemplaza? Aquí está el problema con NAT. Si hubiera un campo extra en el encabezado IP, ese campo podría usarse para guardar el registro del emisor real, pero sólo queda 1 bit sin usar. En principio, podría crearse una nueva opción para mantener la verdadera dirección de origen, pero para ello habría que cambiar el código IP en todas las máquinas de todo Internet para manejar la nueva opción. Ésta no es una alternativa prometedora para un arreglo rápido.

Lo que realmente pasa es lo siguiente. Los diseñadores de NAT observaron que la mayoría de los paquetes IP llevan cargas útiles de TCP o UDP. Cuando estudiemos TCP y UDP en el capítulo 6, veremos que los dos tienen encabezados que contienen un puerto de origen y un puerto de destino. Más adelante explicaremos sólo los puertos TCP, pero esa misma explicación es válida para los puertos UDP. Los puertos son enteros de 16 bits que indican dónde empieza y dónde acaba la conexión TCP. Estos puertos proporcionan el campo requerido para hacer que NAT funcione.

Cuando un proceso desea establecer una conexión TCP con un proceso remoto, se conecta a un puerto TCP sin usar en su propia máquina. Éste se conoce como **puerto de origen** y le indica al código TCP

dónde enviar los paquetes entrantes que pertenecen a esta conexión. El proceso también proporciona un **puerto de destino** para indicar a quién se deben dar los paquetes en el lado remoto. Los puertos 0-1023 se reservan para los servicios conocidos. Por ejemplo, 80 es el puerto usado por los servidores web, para que los clientes remotos puedan localizarlos. Cada mensaje TCP de salida contiene un puerto de origen y uno de destino. En conjunto, estos puertos sirven para identificar los procesos que usan la conexión en ambos extremos.

Una analogía puede aclarar el uso de los puertos. Imagine una compañía con un solo número de teléfono principal. Cuando las personas llaman al número principal, se encuentran con un operador que pregunta qué extensión quieren y entonces los pone en esa extensión. El número principal es análogo a la dirección IP del cliente y las extensiones en ambos extremos son análogas a los puertos. Los puertos son efectivamente 16 bits adicionales de direccionamiento, que identifican qué proceso obtiene cuál paquete entrante.

Si utilizamos el campo *Puerto de origen*, podemos resolver nuestro problema de asignación. Siempre que un paquete de salida entra en la caja NAT, la dirección de origen 10.*x*.*y*.*z* se reemplaza por la verdadera dirección IP del cliente. Además, el campo *Puerto de origen* TCP se reemplaza por un índice en la tabla de traducción de 65 536 entradas de la caja NAT. Esta entrada de la tabla contiene el puerto de origen y la dirección IP originales. Finalmente, las sumas de verificación de los encabezados IP y TCP se recalculan e insertan en el paquete. Es necesario reemplazar el *Puerto de origen*, porque podría ocurrir que ambas conexiones de las máquinas 10.0.0.1 y 10.0.0.2 usaran el puerto 5 000, por ejemplo, así que el *Puerto de origen* no basta por sí solo para identificar el proceso de envío.

Cuando un paquete llega a la caja NAT desde el ISP, el *Puerto de origen* en el encabezado TCP se extrae y utiliza como un índice en la tabla de asignación de la caja NAT. Desde la entrada localizada, la dirección IP interna y el *Puerto de origen* TCP se extraen e insertan en el paquete. Luego, las sumas de verificación de IP y TCP se recalculan e insertan en el paquete. Entonces el paquete se pasa al enrutador del cliente para su entrega normal mediante el uso de la dirección 10.*x*.*y*.*z*.

Aunque este tipo de esquema resuelve el problema, muchos puristas en la comunidad de IP lo consideran a primera vista como una abominación. He aquí algunas de las objeciones sintetizadas brevemente. Primero, NAT viola el modelo arquitectónico de IP, el cual establece que cada dirección IP identifica a una sola máquina en forma única y a nivel mundial. Toda la estructura del software de Internet se basa en este hecho. Con NAT, miles de máquinas pueden usar (y lo hacen) la dirección 10.0.0.1.

Segundo, NAT quebranta el modelo de conectividad de extremo a extremo de Internet, que establece que cualquier host puede enviar un paquete a cualquier otro en cualquier momento dado. Como la asignación en la caja NAT se establece mediante los paquetes de salida, no se pueden aceptar paquetes entrantes sino hasta después de los paquetes de salida. En la práctica, esto significa que un usuario doméstico con NAT puede hacer conexiones TCP/IP a un servidor web remoto, pero un usuario remoto no puede hacer conexiones a un servidor de juego en la red doméstica. Se requiere una configuración especial o técnicas de **NAT transversal** para soportar este tipo de situación.

Tercero, NAT cambia a Internet de una red sin conexión a un tipo especial de red orientada a conexión. El problema es que la caja NAT debe mantener la información (es decir, la asignación) para cada conexión que pasa a través de ella. Hacer que la red mantenga el estado de la conexión es una propiedad de las redes orientadas a conexión, no de las redes sin conexión. Si la caja NAT falla y se pierde su tabla de asignación, se destruirán todas sus conexiones TCP. En ausencia de NAT, un enrutador puede fallar y reiniciarse sin un efecto a largo plazo sobre las conexiones TCP. El proceso de envío apenas queda fuera unos segundos y retransmite todos los paquetes cuya recepción no se haya confirmado. Con NAT, Internet es tan vulnerable como una red de circuitos conmutados.

Cuarto, NAT viola la regla más fundamental de los protocolos distribuidos en capas: la capa k no puede hacer ninguna suposición acerca de lo que la capa $k + 1$ ha puesto en el campo de carga útil. Este principio básico está ahí para que las capas sigan siendo independientes. Si TCP se actualiza después a

TCP-2, con un diseño de encabezado diferente (por ejemplo, puertos de 32 bits), NAT fallará. Toda la idea de los protocolos distribuidos en capas es asegurar que los cambios en una capa no requieran cambios en otras capas. NAT destruye esta independencia.

Quinto, en Internet no se exige que los procesos utilicen TCP o UDP. Si un usuario en la máquina *A* decide usar algún nuevo protocolo de transporte para hablar con un usuario en la máquina *B* (por ejemplo, para una aplicación multimedia), la introducción de una caja NAT hará que la aplicación falle, ya que la caja NAT no podrá localizar el *Puerto de origen* TCP de forma correcta.

Un sexto problema relacionado consiste en que algunas aplicaciones usan múltiples conexiones TCP/IP o puertos UDP en las formas prescritas. Por ejemplo, el **FTP** (**Protocolo de Transferencia de Archivos estándar**, del inglés *File Transfer Protocol*) inserta direcciones IP en el cuerpo del paquete para que el receptor las extraiga y utilice. Como NAT no sabe nada sobre estos arreglos, no puede reescribir las direcciones IP ni justificarlas de alguna otra forma. Esta falta de entendimiento significa que FTP y otras aplicaciones como el protocolo de telefonía de Internet H.323 (que estudiaremos en el capítulo 7) pueden fallar en presencia de NAT, a menos que se tomen precauciones especiales. A menudo es posible arreglar NAT para estos casos, pero no es una buena idea tener que arreglar el código en la caja NAT cada vez que surge una nueva aplicación.

Por último, debido a que el campo *Puerto de origen* de TCP es de 16 bits, a lo sumo se pueden asignar 65 536 máquinas a una dirección IP. En realidad, el número es ligeramente menor porque los primeros 4 096 puertos se reservan para usos especiales. Pero si hay varias direcciones IP disponibles, cada una puede manejar hasta 61 440 máquinas.

En el RFC 2 993 se explican éstos y otros problemas con NAT. A pesar de estas cuestiones, NAT se utiliza mucho en la práctica, en especial para las redes domésticas y de negocios pequeños, como la única técnica conveniente para lidiar con la escasez de direcciones IP. Se ha visto envuelta con los firewalls y la privacidad debido a que bloquea de manera predeterminada los paquetes entrantes no solicitados. Por esta razón es muy poco probable que desaparezca, incluso aunque el IPv6 se implemente de manera amplia.

5.6.3 IP versión 6

IP se ha utilizado intensivamente durante décadas. Ha trabajado muy bien, como lo demuestra el crecimiento exponencial de Internet. Por desgracia, IP se ha convertido en una víctima de su propia popularidad: está cerca de quedarse sin direcciones. Aun con las técnicas CIDR y NAT que utilizan las direcciones con más moderación, se espera que la ICANN asigne las últimas direcciones IPv4 antes de que termine el año 2012. Este inminente desastre se reconoció hace casi más de dos décadas y fue motivo de grandes discusiones y controversias dentro de la comunidad de Internet, para ver qué hacer al respecto.

En esta sección describiremos tanto el problema como varias soluciones propuestas. La única solución a largo plazo es cambiar a direcciones más grandes. **IPv6** (**IP versión 6**) es un diseño de reemplazo que hace precisamente eso. Puesto que utiliza direcciones de 128 bits, es muy poco probable que se vayan a agotar en un futuro previsible. Sin embargo, IPv6 ha demostrado ser muy difícil de implementar. Es un protocolo de capa de red diferente que en realidad no congenia internamente con IPv4, a pesar de tantas similitudes. Además, las empresas y los usuarios en realidad no están seguros de por qué querrían IPv6 en cualquier caso. El resultado es que IPv6 se implementa y utiliza sólo en una pequeña fracción de Internet (se estima un 1%), a pesar de haber sido un estándar de Internet desde 1998. Los siguientes años serán interesantes, a medida que se vayan asignando las pocas direcciones IPv4 restantes. ¿Empezarán las personas a subastar sus direcciones IPv4 en eBay? ¿Surgirá un mercado negro para su compra/venta? Nadie lo sabe.

Además de los problemas de las direcciones, hay otras cuestiones que amenazan en segundo plano. En sus primeros años, Internet era utilizada por universidades, industrias de alta tecnología y el gobierno

de Estados Unidos. (en especial, el Departamento de Defensa). Con la explosión de interés sobre Internet que empezó a mediados de la década de 1990, un grupo distinto de personas empezó a utilizarla, a menudo con distintos requerimientos. En primer lugar, muchas personas con teléfonos inteligentes la utilizan para mantenerse en contacto con sus bases. En segundo lugar, con la convergencia inminente de las industrias de las computadoras, la comunicación y el entretenimiento, tal vez no pase mucho tiempo antes de que todos los teléfonos y televisiones en el mundo sean un nodo de Internet, con lo cual 1000 millones de máquinas se utilizarán para el audio y video bajo demanda. Bajo estas circunstancias, era evidente que IP tenía que evolucionar y hacerse más flexible.

Al ver estos problemas en el horizonte, en 1990 la IETF empezó a trabajar sobre una nueva versión de IP, una que nunca se quedara sin direcciones, que resolviera una variedad de problemas adicionales y también fuera más flexible y eficiente. Sus principales objetivos eran:

1. Soportar miles de millones de hosts, incluso con una asignación de direcciones ineficiente.
2. Reducir el tamaño de las tablas de enrutamiento.
3. Simplificar el protocolo para permitir a los enrutadores procesar los paquetes con más rapidez.
4. Proporcionar mayor seguridad (autentificación y privacidad).
5. Poner más atención en cuanto al tipo de servicio, en especial para los datos en tiempo real.
6. Ayudar a la multidifusión al permitir la especificación de alcances.
7. Permitir que un host deambule libremente sin tener que cambiar su dirección.
8. Permitir que el protocolo evolucione en el futuro.
9. Permitir que el protocolo viejo y el nuevo coexistan durante años.

El diseño de IPv6 presentaba una gran oportunidad para mejorar todas las características en IPv4 que estaban muy lejos de cumplir con lo que ahora se necesitaba. Para desarrollar un protocolo que cumpliera con todos esos requerimientos, la IETF emitió una llamada para propuestas y discusión en el RFC 1550. Al principio se recibieron 21 respuestas. Para diciembre de 1992, había siete propuestas serias en la mesa, que variaban desde hacer pequeñas correcciones a IP, hasta descartarlo por completo y reemplazarlo con un protocolo por completo distinto.

Una propuesta fue la de operar TCP sobre CLNP, el protocolo de capa de red diseñado para OSI. Con sus direcciones de 160 bits, CLNP hubiera proporcionado suficiente espacio de direcciones por un tiempo indefinido, ya que era capaz de dar a cada una de las moléculas de agua en los océanos suficientes direcciones (aproximadamente 2^5) para establecer una pequeña red. Esta opción también hubiera unificado dos de los principales protocolos de capa de red. Sin embargo, muchas personas sentían que esto hubiera sido como admitir que algo en el mundo de OSI se había hecho bien en realidad, una declaración considerada como políticamente incorrecta en los círculos de Internet. CLNP estaba modelado en forma muy parecida a IP, por lo que en realidad los dos no son tan diferentes. De hecho, el protocolo que se eligió en última instancia difiere más de IP que CLNP. Otro golpe contra CLNP era su defectuoso soporte para los tipos de servicios, algo que se requería para transmitir la multimedia con eficiencia.

Tres de las mejores propuestas se publicaron en *IEEE Network* (Deering, 1993; Francis, 1993; Katz y Ford, 1993). Después de mucha discusión, revisión y competencia por posición, se seleccionó una versión combinada modificada de las propuestas de Deering y Francis, ahora conocida como **SIPP** (**Protocolo Simple de Internet Mejorado**, del inglés *Simple Internet Protocol Plus*), designado también como **IPv6**.

IPv6 cumple bastante bien con los objetivos de la IETF. Mantiene las buenas características de IP, descarta o desenfatiza las malas y agrega nuevas en donde se requiere. En general, IPv6 no es compatible con IPv4, pero es compatible con los otros protocolos auxiliares de Internet, incluyendo TCP, UDP, ICMP, IGMP, OSPF, BGP y DNS, con pequeñas modificaciones requeridas para lidiar con las direcciones más

largas. A continuación describiremos las principales características de IPv6. Encontrará más información sobre este protocolo en los RFC 2460 al 2 466.

Primero que nada, IPv6 tiene direcciones más largas que IPv4. Son de 128 bits, lo cual resuelve el problema deseado: proporcionar un suministro efectivamente ilimitado de direcciones de Internet. En breve hablaremos más sobre las direcciones.

La segunda mejora importante de IPv6 es la simplificación del encabezado. Sólo contiene siete campos (en comparación con los 13 de IPv4). Este cambio permite a los enrutadores procesar los paquetes con más rapidez y, por ende, mejora la velocidad de transmisión real y el retardo. También hablaremos en breve sobre el encabezado.

La tercera mejora importante es un soporte mejorado para las opciones. Este cambio era esencial con el nuevo encabezado, ya que los campos que antes eran requeridos ahora son opcionales (debido a que no se utilizan con tanta frecuencia). Además, la forma en que se representan las opciones es distinta y así es más fácil para los enrutadores omitir las porciones que no están destinadas para ellos. Esta característica agiliza el tiempo de procesamiento de los paquetes.

Una cuarta área en la que IPv6 representa un gran avance es la seguridad. La IETF tuvo suficiente con las historias en los periódicos sobre los niños precoces de 12 años que usaban sus computadoras personales para irrumpir en los bancos y las bases militares a través de Internet. Hubo una fuerte sensación de tener que hacer algo para mejorar la seguridad. La autentificación y la privacidad son características clave del nuevo IP. Sin embargo, después se modernizaron para el IPv4, por lo que en el área de la seguridad las diferencias ya no son tan grandes.

Por último, se puso más énfasis en la calidad del servicio. Se han hecho varios esfuerzos carentes de entusiasmo para mejorar la QoS en el pasado, pero ahora con el crecimiento de la multimedia en Internet, la sensación de urgencia es mayor.

El encabezado principal de IPv6

En la figura 5-56 se muestra el encabezado de IPv6. El campo *Versión* siempre es 6 para IPv6 (y 4 para IPv4). Durante el periodo de transición de IPv4, que ya se ha tardado más de una década, los enrutadores podrán examinar este campo para saber qué tipo de paquete tienen. Como observación adicional, para hacer esta prueba se desperdician unas cuantas instrucciones en la ruta crítica, dado que el encabezado de enlace de datos indica por lo general el protocolo de red para demultiplexar, por lo que tal vez algunos enrutadores omitan la verificación. Por ejemplo, el campo *Tipo* de Ethernet tiene distintos valores para indicar una carga útil de IPv4 o IPv6. Las discusiones entre los campos “Hacerlo bien” y “Hacerlo rápido” sin duda serán extensas y vigorosas.

El campo *Servicios diferenciados* (originalmente conocido como *Clase de tráfico*) se utiliza para distinguir la clase de servicio para los paquetes con distintos requerimientos de entrega en tiempo real. Se utiliza con la arquitectura de servicio diferenciado para la calidad del servicio, de la misma forma que el campo con el mismo nombre en el paquete IPv4. Además, los 2 bits de menor orden se usan para señalar las indicaciones explícitas de congestión, de nuevo en la misma forma que IPv4.

El campo *Etiqueta de flujo* proporciona el medio para que un origen y un destino marquen grupos de paquetes que tengan los mismos requerimientos y que la red deba tratar de la misma forma, para formar una seudoconexión. Por ejemplo, un flujo de paquetes de un proceso en cierto host de origen dirigido a cierto proceso en un host de destino puede tener requisitos muy estrictos de retardo y, por lo tanto, necesitar un ancho de banda reservado. El flujo se puede establecer por adelantado, dándole un identificador. Cuando aparece un paquete con una *Etiqueta de flujo* diferente de cero, todos los enrutadores pueden buscarla en sus tablas internas para ver el tipo de tratamiento especial que requiere. En efecto, los flujos son un intento de tener lo mejor de ambos mundos: la flexibilidad de una red de datagramas y las garantías de una red de circuitos virtuales.

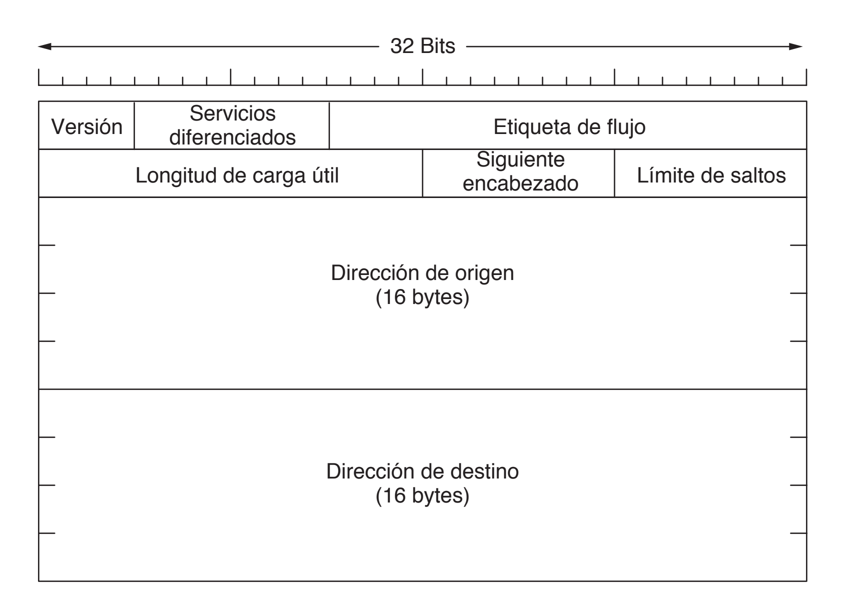


Figura 5-56. El encabezado fijo de IPv6 (requerido).

Para fines de la calidad del servicio, cada flujo está designado con base en la dirección de origen, la dirección de destino y el número de flujo. Este diseño significa que pueden estar activos hasta 2^{20} flujos al mismo tiempo entre un par dado de direcciones IP. También significa que, incluso si dos flujos provenientes de hosts diferentes pero con la misma etiqueta de flujo pasan por el mismo enrutador, éste será capaz de distinguirlos mediante las direcciones de origen y de destino. Lo ideal es que las etiquetas de flujo se escojan al azar, en vez de asignarlas de manera secuencial comenzando por el 1, de modo que los enrutadores tengan que usar *hashing*.

El campo *Longitud de carga útil* indica cuántos bytes van después del encabezado de 40 bytes de la figura 5-56. El nombre se cambió de *Longitud total* en el IPv4 porque el significado cambió ligeramente: los 40 bytes del encabezado ya no se cuentan como parte de la longitud (como antes). Este cambio significa que ahora la carga útil puede ser de 65 535 bytes en vez de sólo 65 515 bytes.

El campo *Siguiente encabezado* revela el secreto. La razón por la que pudo simplificarse el encabezado es que puede haber encabezados adicionales (opcionales) de extensión. Este campo indica cuál de los seis encabezados de extensión (en la actualidad), siguen de éste, en caso de que los haya. Si este encabezado es el último encabezado de IP, el campo *Siguiente encabezado* indica el protocolo de transporte (por ejemplo, TCP, UDP) al que se entregará el paquete.

El campo *Límite de saltos* se usa para evitar que los paquetes vivan eternamente. En la práctica es igual al campo *Tiempo de vida* del IPv4; esto es, un campo que se disminuye en cada salto. En teoría, en el IPv4 era un tiempo en segundos, pero ningún enrutador lo usaba de esa manera, por lo que se cambió el nombre para reflejar la manera en que se usa realmente.

Luego vienen los campos *Dirección de origen* y *Dirección de destino*. La propuesta original de Deering, SIP, usaba direcciones de 8 bytes pero durante el periodo de revisión muchas personas sintieron que, con direcciones de 8 bytes, en unas cuantas décadas el IPv6 se quedaría sin direcciones, mientras que con direcciones de 16 bytes nunca se agotarían. Otros argumentaban que 16 bytes era demasiado, mientras que unos más estaban a favor de usar direcciones de 20 bytes para hacerlas compatibles con el protocolo de datagramas de OSI. Para colmo, otro grupo quería direcciones de tamaño variable. Tras mucho debate y más de unas cuantas palabras inmencionables en un libro de texto académico, se decidió que la mejor medida sería una dirección de 16 bytes de longitud fija.

Se ha desarrollado una nueva notación para escribir direcciones de 16 bytes, las cuales se escriben como ocho grupos de cuatro dígitos hexadecimales, separando los grupos por dos puntos, como el siguiente ejemplo:

8000:0000:0000:0000:0123:4567:89AB:CDEF

Ya que muchas direcciones tendrán muchos ceros en ellas, se han autorizado tres optimizaciones. Primero, se pueden omitir los ceros a la izquierda dentro de un grupo, por lo que es posible escribir 0123 como 123. Segundo, se pueden reemplazar uno o más grupos de 16 bits cero por un par de signos de dos puntos. Por lo tanto, la dirección anterior se vuelve ahora

8000::123:4567:89AB:CDEF

Por último, las direcciones IPv4 se pueden escribir como un par de signos de dos puntos y un número decimal anterior separado por puntos, por ejemplo:

::192.31.20.46

Tal vez no sea necesario ser tan explícitos al respecto, pero hay muchas direcciones de 16 bytes. Específicamente hay 2^{128} de ellas, lo que aproximadamente equivale a 3×10^{38}. Si la Tierra completa, incluidos los océanos, estuviera cubierta de computadoras, el IPv6 permitiría 7×10^{23} direcciones IP por metro cuadrado. Los estudiantes de química notarán que este número es mayor que el número de Avogadro. Aunque no fue la intención darle a cada molécula de la superficie terrestre su propia dirección IP, no estamos lejos de ello.

En la práctica, el espacio de direcciones no se usará de manera eficiente, justo igual que el espacio de direcciones de números telefónicos (el código de área de Manhattan, 212, está prácticamente lleno, pero el de Wyoming, 307, está casi vacío). En el RFC 3194, Durand y Huitema calcularon que, si se usa la asignación de números telefónicos como guía, hasta en la situación más pesimista habrá más de 1000 direcciones IP por metro cuadrado de la superficie terrestre (tierra y agua). En cualquier situación probable, habrá billones de ellas por metro cuadrado. En pocas palabras, parece poco probable que se vayan a agotar en el futuro previsible.

Para fines instructivos, podemos comparar el encabezado de IPv4 (figura 5-46) con el de IPv6 (figura 5-56) para ver lo que se ha dejado fuera del IPv6. El campo *IHL* se fue porque el encabezado de IPv6 tiene una longitud fija. El campo *Protocolo* se retiró porque el campo *Siguiente encabezado* indica lo que sigue después del último encabezado IP (por ejemplo, un segmento UDP o TCP).

Se eliminaron todos los campos relacionados con la fragmentación, puesto que el IPv6 tiene un enfoque distinto en cuanto a la fragmentación. Para empezar, todos los hosts que cumplen con el IPv6 deben determinar en forma dinámica el tamaño de paquete a usar. Para ello utilizan el procedimiento de descubrimiento de MTU de la ruta que describimos en la sección 5.5.5. En resumen, cuando un host envía un paquete IPv6 demasiado grande, en vez de fragmentarlo, el enrutador que no puede reenviarlo descarta el paquete y envía un mensaje de error de vuelta al host emisor. Este mensaje indica al host que divida todos los paquetes futuros para ese destino. Hacer que el host envíe paquetes del tamaño correcto en primer lugar es, en última instancia, mucho más eficiente que hacer que los enrutadores los fragmenten sobre la marcha. Asimismo, el tamaño mínimo de paquete se incrementó de 576 a 1280 bytes para permitir 1024 bytes de datos y muchos encabezados.

Por último, el campo *Suma de verificación* desapareció porque al calcularlo se reduce en gran medida el desempeño. Con las redes confiables de hoy, además del hecho de que la capa de enlace de datos y las capas de transporte por lo general tienen sus propias sumas de verificación, la ventaja de tener otra suma de verificación no valía el costo de desempeño que generaba. Al quitar estas características, el resultado es un protocolo de capa de red compacto y sólido. Por lo tanto, con este diseño se cumplió la meta del IPv6 (un protocolo rápido y flexible con bastante espacio de direcciones).

Encabezados de extensión

Como de todas formas ocasionalmente se requieren algunos de los campos faltantes del IPv4, el IPv6 introdujo el concepto de **encabezados de extensión** (opcionales). Estos encabezados se pueden usar para proporcionar información adicional, pero codificada de una manera eficiente. Hay seis tipos de encabezados de extensión definidos en la actualidad, los cuales se listan en la figura 5-57. Todos son opcionales, pero si hay más de uno presente, deben aparecer justo después del encabezado fijo y de preferencia en el orden listado.

Encabezado de extensión	Descripción
Opciones salto por salto.	Información diversa para los enrutadores.
Opciones de destino.	Información adicional para el destino.
Enrutamiento.	Lista informal de los enrutadores a visitar.
Fragmentación.	Manejo de fragmentos de datagramas.
Autentificación.	Verificación de la identidad del emisor.
Carga útil de seguridad cifrada.	Información sobre el contenido cifrado.

Figura 5-57. Encabezados de extensión de IPv6.

Algunos de los encabezados tienen un formato fijo; otros contienen un número variable de opciones de longitud variable. En éstos, cada elemento está codificado como una tupla (*Tipo*, *Longitud*, *Valor*). El *Tipo* es un campo de 1 byte que indica la opción de la que se trata. Los valores de *Tipo* se han escogido de modo que los 2 primeros bits indiquen a los enrutadores, que no saben cómo procesar la opción, lo que tienen que hacer. Las posibilidades son: omitir la opción, descartar el paquete, descartar el paquete y enviar de vuelta un paquete ICMP, y descartar el paquete pero no enviar paquetes ICMP para direcciones de multidifusión (para evitar que un paquete de multidifusión malo genere millones de informes ICMP).

La *Longitud* también es un campo de 1 byte, e indica la longitud del valor (0 a 255 bytes). El *Valor* es cualquier información requerida de hasta 255 bytes.

El encabezado de salto por salto se usa para la información que deben examinar todos los enrutadores a lo largo de la ruta. Hasta ahora, se ha definido una opción: manejo de datagramas de más de 64K. El formato de este encabezado se muestra en la figura 5-58. Cuando se utiliza, el campo *Longitud de carga útil* del encabezado fijo se establece a 0.

<table>
<tr><td>Siguiente encabezado</td><td>0</td><td>194</td><td>4</td></tr>
<tr><td colspan="4">Longitud de carga útil de tamaño extra</td></tr>
</table>

Figura 5-58. El encabezado de extensión salto por salto para datagramas grandes (jumbogramas).

Como todos los encabezados de extensión, éste comienza con un byte que indica el tipo de encabezado que sigue. A este byte le sigue uno que indica la longitud del encabezado salto por salto en bytes, excluyendo los primeros 8 bytes, que son obligatorios. Todas las extensiones empiezan de esta manera.

Los siguientes 2 bytes indican que esta opción define el tamaño del datagrama (código 194) y que el tamaño es un número de 4 bytes. Los últimos 4 bytes indican el tamaño del datagrama. No se permiten tamaños menores a 65 536 bytes, pues de lo contrario el primer enrutador descartará el paquete y devolverá un mensaje ICMP de error. Los datagramas que usan este encabezado de extensión se llaman **jumbogramas**. El uso de jumbogramas es importante para las aplicaciones de supercomputadoras que deben transferir con eficiencia gigabytes de datos a través de Internet.

El encabezado de opciones de destino está reservado para campos que sólo se necesitan interpretar en el host de destino. En la versión inicial de IPv6, las únicas opciones definidas son las opciones nulas para rellenar este encabezado a un múltiplo de 8 bytes, por lo que en un principio no se usará. Se incluyó para asegurarse que el nuevo enrutamiento y el software del host pudieran manejarlo, en caso de que algún día alguien pensara en una opción de destino.

El encabezado de enrutamiento lista uno o más enrutadores que deben visitarse en el camino hacia el destino. Es muy similar al enrutamiento de origen libre del IPv4 en cuanto a que todas las direcciones listadas se deben visitar en orden, pero también se pueden visitar otros enrutadores no listados que se encuentren dentro del trayecto. El formato del encabezado de enrutamiento se muestra en la figura 5-59.

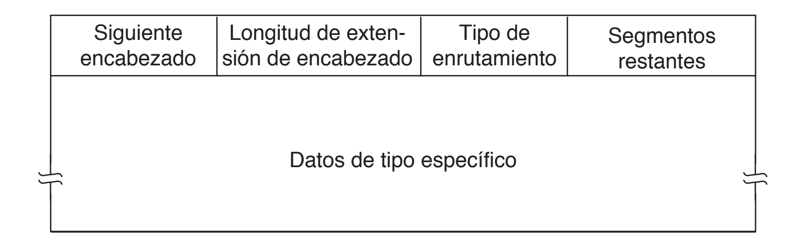


Figura 5-59. El encabezado de extensión para enrutamiento.

Los primeros 4 bytes del encabezado de extensión de enrutamiento contienen cuatro enteros de 1 byte. Anteriormente describimos los campos *Siguiente encabezado* y *Longitud de extensión del encabezado*. El campo *Tipo de enrutamiento* proporciona el formato del resto del encabezado. El tipo 0 indica que una palabra reservada de 32 bits va después de la primera palabra, seguida por cierto número de direcciones de IPv6. Pueden inventarse otros tipos en el futuro según se necesite. Por último, el campo *Segmentos restantes* registra cuántas direcciones de la lista no se han visitado todavía. Se decrementa cada vez que se visita una de ellas. Cuando llega a 0, el paquete está por su cuenta, sin más guía sobre qué ruta seguir. Por lo general, a estas alturas está tan cerca del destino que la mejor ruta es obvia.

El encabezado de fragmentación maneja ésta de una manera parecida a la del IPv4. El encabezado contiene el identificador del datagrama, el número de fragmento y un bit que indica si seguirán más fragmentos. A diferencia del IPv4, en el IPv6 sólo el host de origen puede fragmentar un paquete. Los enrutadores a lo largo del camino no pueden hacerlo. Este cambio representa un rompimiento filosófico con el IP original, aunque se mantiene a la par con la práctica habitual del IPv4. Además, simplifica el trabajo del enrutador y acelera el proceso de enrutamiento. Como se mencionó antes, si un enrutador confronta un paquete demasiado grande, lo descarta y devuelve un paquete de error ICMP al origen. Esta información permite que el host de origen fragmente el paquete en piezas más pequeñas mediante el uso de este encabezado y lo intente de nuevo.

El encabezado de autentificación proporciona un mecanismo mediante el cual el receptor de un paquete puede estar seguro de quién lo envió. La carga útil de seguridad de cifrado hace posible cifrar el contenido de un paquete, de modo que sólo el receptor pretendido pueda leerlo. Estos encabezados usan las técnicas criptográficas que describiremos en el capítulo 8 para lograr su cometido.

Controversias

Dados el proceso de diseño abierto y las fuertes opiniones de muchas de las personas participantes, no debería sorprender que muchas decisiones tomadas para el IPv6 fueran tema de fuertes controversias. Resumiremos a continuación algunas de ellas. Para los detalles, vea los RFC.

Ya hemos mencionado el argumento sobre la longitud de las direcciones. El resultado fue una solución intermedia: direcciones de 16 bytes de longitud fija.

Surgió otra pelea sobre la longitud del campo *Límite de saltos*. Una parte sentía que limitar la cantidad máxima de saltos a 255 (implícita al usar un campo de 8 bits) era un grave error. A fin de cuentas, hoy son comunes las rutas de 32 saltos, y en 10 años más podrán ser comunes rutas mucho más extensas. Esta gente argumentaba que usar una dirección con un tamaño enorme era ir demasiado lejos, pero que el uso de una pequeña cuenta de saltos era tener una visión miope. Desde su punto de vista, el peor pecado que puede cometer un científico de la computación es proporcionar muy pocos bits en algún lugar.

La respuesta fue que podían existir buenas razones para aumentar todos los campos, lo que llevaría a un encabezado congestionado. Además, la función del campo *Límite de saltos* es evitar que los paquetes vaguen por mucho tiempo, por lo cual 65 535 saltos son demasiados. Por último, a medida que crezca Internet se construirán cada vez más enlaces de larga distancia, mediante lo cual será posible ir de un país a otro en media docena de saltos, cuando mucho. Si se requieren más de 125 saltos para llegar del origen y el destino a sus puertas de enlace internacionales, algo está mal con las redes troncales nacionales. Los de 8 bits ganaron esta partida.

Otra "papa caliente" fue el tamaño máximo de paquete. La comunidad de las supercomputadoras quería paquetes de más de 64 KB. Cuando una supercomputadora comienza a transferir, el asunto realmente va en serio, y no quiere que se le interrumpa cada 64 KB. El argumento en contra de los paquetes grandes es que, si un paquete de 1 MB llega a una línea T1 de 1.5 Mbps, ese paquete bloqueará la línea durante más de 5 segundos, produciendo un retardo muy notorio para los usuarios interactivos que comparten la línea. Se llegó a un punto medio: los paquetes normales se limitan a 64 KB, pero se puede usar el encabezado de extensión de salto por salto para permitir los jumbogramas.

Un tercer tema candente fue la desaparición de la suma de verificación del IPv4. Para algunas personas esto representa algo parecido a quitarle los frenos a un automóvil. Hacerlo ciertamente aligera al automóvil y, por lo tanto, puede ir más rápido pero, de ocurrir un evento inesperado, tendremos problemas.

El argumento en contra de las sumas de verificación fue que cualquier aplicación a la que de verdad le importe la integridad de sus datos de todos modos tiene que tener una suma de verificación en la capa de transporte, por lo que tener otra en el IP (además de la suma de verificación de la capa de enlace de datos) es un exceso. Además, la experiencia mostraba que calcular la suma de verificación del IP representaba un gasto considerable en el IPv4. El bando en contra de la suma de verificación ganó esta partida, por lo que el IPv6 no tiene una suma de verificación.

Los hosts móviles también fueron tema de contienda. Si una computadora portátil vuela al otro lado del mundo, ¿puede continuar operando ahí con la misma dirección IPv6, o tiene que usar un esquema con agentes de base? Algunas personas querían incluir soporte explícito para hosts móviles en el IPv6. Este esfuerzo falló cuando no se pudo generar consenso para ninguna propuesta específica.

Es probable que la batalla más importante fue sobre la seguridad. Todos estaban de acuerdo con que era esencial. La guerra fue sobre dónde y cuándo usarla. Primero dónde. El argumento a favor de ponerla en la capa de red es que entonces se vuelve un servicio estándar que todas las aplicaciones pueden usar sin necesidad de planearlo por adelantado. El argumento en contra es que las aplicaciones realmente seguras por lo general no quieren nada menos que el cifrado de terminal a terminal, donde la aplicación de origen hace el cifrado y la aplicación de destino lo deshace. Con cualquier otra cosa menos, el usuario está a merced de implementaciones de capa de red con fallas potenciales, sobre las que no tiene control. La respuesta

a este argumento es que tales aplicaciones simplemente pueden abstenerse de usar las características de seguridad del IP y encargarse ellas mismas del asunto. La réplica a esto es que la gente que no confía en que la red lo haga bien no quiere pagar el precio de implementaciones de IP lentas y estorbosas que tengan esta capacidad, aun si está deshabilitada.

Otro aspecto sobre dónde poner la seguridad se relaciona con el hecho de que muchos países (pero no todos) tienen leyes de exportación muy estrictas en lo referente al cifrado. Algunos, en especial Francia e Irak, también restringen mucho su uso doméstico, de manera que la gente no pueda ocultar secretos al gobierno. Como resultado, cualquier implementación de IP que use un sistema de cifrado lo bastante robusto como para tener algún valor no podría exportarse de Estados Unidos (y de muchos otros países) a clientes mundiales. Tener que mantener dos conjuntos de software, uno para uso doméstico y otro para exportación, es algo a lo que la mayoría de los distribuidores de computadoras se oponen enérgicamente.

Un punto donde no hubo controversia es que nadie espera que la Internet IPv4 se apague un domingo por la mañana y reinicie como Internet IPv6 la mañana del lunes. En cambio, se convertirán "islas" de IPv6, que en un principio se comunicarán a través de túneles como vimos en la sección 5.5.3. A medida que crezcan las islas IPv6, se integrarán a islas más grandes. Tarde o temprano todas las islas se integrarán, y la Internet se habrá convertido por completo.

Por lo menos ése era el plan. La implementación ha demostrado ser el talón de Aquiles de IPv6. Se sigue usando muy poco, aun cuando todos los sistemas operativos principales tienen soporte completo para IPv6. La mayoría de las implementaciones son nuevos casos en los que un operador de red (por ejemplo, un operador de telefonía móvil) necesita una gran cantidad de direcciones IP. Se han definido muchas estrategias para ayudar a facilitar la transición. Entre éstas hay formas de configurar de manera automática los túneles que transportan IPv6 sobre la Internet IPv4, y formas de que los hosts encuentren de manera automática las terminales de los túneles. Los hosts de pila dual tienen una implementación IPv4 y otra IPv6, de modo que puedan seleccionar qué protocolo usar dependiendo del destino del paquete. Estas estrategias optimizarán la implementación substancial que parece inevitable cuando se agoten las direcciones IPv4. Para obtener más información sobre IPv6, consulte a Davies (2008).

5.6.4 Protocolos de control en Internet

Además el IP que se utiliza para la transferencia de datos, Internet tiene varios protocolos de control complementarios que se utilizan en la capa de red, como ICMP, ARP y DHCP. En esta sección los analizaremos por separado y describiremos las versiones que corresponden a IPv4, ya que son los protocolos de uso común. ICMP y DHCP tienen versiones similares para IPv6; el equivalente de ARP se llama **NDP** (**Protocolo de Descubrimiento de Vecino**, del inglés *Neighbor Discovery Protocol*) para IPv6.

ICMP: Protocolo de Mensajes de Control en Internet

Los enrutadores supervisan muy de cerca el funcionamiento de Internet. Cuando ocurre algo inesperado durante el procesamiento de un paquete en un enrutador, **ICMP** (**Protocolo de Mensajes de Control en Internet**, del inglés *Internet Control Message Protocol*) informa sobre el evento al emisor. ICMP también se utiliza para probar Internet. Hay definidos alrededor de una docena de tipos de mensajes ICMP, cada uno de los cuales se transporta encapsulado en un paquete IP. Los más importantes se listan en la figura 5-60.

El mensaje *DESTINATION UNREACHABLE* (DESTINO INACCESIBLE) se usa cuando el enrutador no puede localizar el destino o cuando un paquete con el bit *DF* no puede entregarse porque hay una red de "paquetes pequeños" que se interpone en el camino.

Tipo de mensaje	Descripción
Destination unreachable (Destino inaccesible).	No se pudo entregar el paquete.
Time exceeded (Tiempo excedido).	El tiempo de vida llegó a cero.
Parameter problem (Problema de parámetros).	Campo de encabezado inválido.
Source quench (Fuente disminuida).	Paquete regulador.
Redirect (Redireccionar).	Enseña a un enrutador la geografía.
Echo and echo reply (Eco y respuesta de eco).	Verifica si una máquina está viva.
Timestamp request/reply (Estampa de tiempo, Petición/respuesta).	Igual que solicitud de eco, pero con marca de tiempo.
Router advertisement/solicitation (Enrutamiento anuncio/solicitud).	Busca un enrutador cercano.

Figura 5-60. Los principales tipos de mensajes ICMP.

El mensaje *TIME EXCEEDED* (TIEMPO EXCEDIDO) se envía al descartar un paquete porque su contador *Ttl* (*Tiempo de vida*) ha llegado a cero. Este evento es un síntoma de que los paquetes se están repitiendo o que los valores establecidos en el contador son muy bajos.

Un uso inteligente de este mensaje de error es la herramienta **traceroute** desarrollada por Van Jacobson en 1987. Traceroute encuentra los enrutadores a lo largo de la ruta desde el host hasta una dirección IP de destino. Encuentra esta información sin ningún tipo de soporte de red privilegiado. Este método simplemente envía una secuencia de paquetes al destino, primero con un TtL de 1, después con un TtL de 2, 3 y así en lo sucesivo. Los contadores en estos paquetes llegarán a cero en los enrutadores sucesivos a lo largo de la ruta. Cada uno de estos enrutadores enviará obedientemente un mensaje *TIME EXCEEDED* de vuelta al host. A partir de estos mensajes el host puede determinar las direcciones IP de los enrutadores a lo largo de la ruta, así como mantener estadísticas y tiempos en ciertas partes de la ruta. No es para lo que estaba destinado el mensaje *TIME EXCEEDED*, pero tal vez sea la herramienta de depuración de red más útil de todos los tiempos.

El mensaje *PARAMETER PROBLEM* (PROBLEMAS DE PARÁMETROS) indica que se ha descubierto un valor ilegal en un campo de encabezado. Este problema indica un error en el software de IP del host emisor, o tal vez en el software de un enrutador por el que se transita.

El mensaje *SOURCE QUENCH* (FUENTE DISMINUIDA) se utilizaba hace tiempo para regular a los hosts que estaban enviando demasiados paquetes. Se esperaba que cuando un host recibiera este mensaje, redujera la velocidad. En la actualidad raras veces se usa pues cuando ocurre una congestión, estos paquetes tienden a agravar más la situación y no está claro cómo responderles. Ahora, el control de congestión en Internet se hace sobre todo en la capa de transporte, en donde se utilizan las pérdidas de paquetes como señales de congestión; lo estudiaremos con detalle en el capítulo 6.

El mensaje *REDIRECT* (REDIRECCIONAR) se usa cuando un enrutador se percata de que un paquete parece estar mal enrutado. Lo utiliza el enrutador para avisar al host emisor que se actualice con una mejor ruta.

Los mensajes *ECHO* (ECO) y *ECHO REPLY* (RESPUESTA DE ECO) se utilizan para ver si un destino dado es alcanzable y está vivo. Se espera que el destino envíe de vuelta un mensaje *ECHO REPLY* luego de recibir el mensaje *ECHO*. Estos mensajes se utilizan en la herramienta **ping** que verifica si un host está activo en Internet.

Los mensajes *TIMESTAMP REQUEST* (PETICIÓN DE ESTAMPA DE TIEMPO) y *TIMESTAMP REPLY* (RESPUESTA DE ESTAMPA DE TIEMPO) son similares, excepto que el tiempo de llegada del mensaje y el tiempo de salida de la respuesta se registran en ésta. Esta característica se puede usar para medir el desempeño de la red.

Los mensajes *ROUTER ADVERTISEMENT* (ANUNCIO DE ENRUTADOR) y *ROUTER SOLICITATION* (SOLICITUD DE ENRUTADOR) se usan para permitir que los hosts encuentren los enrutadores cercanos. Un host necesita aprender la dirección IP de por lo menos un enrutador para enviar paquetes por la red local.

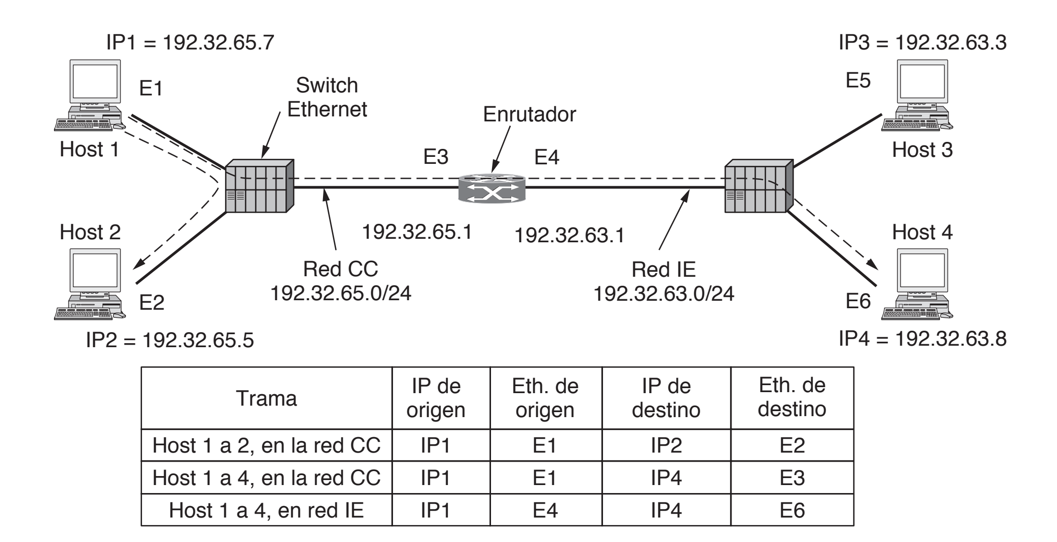
Además de estos mensajes, se han definido otros. La lista en línea se conserva ahora en www.iana.org/assignments/icmp-parameters.

ARP: Protocolo de Resolución de Direcciones

Aunque en Internet cada máquina tiene una o más direcciones IP, en realidad éstas no son suficientes para enviar paquetes. Las NIC (Tarjetas de Interfaz de Red) de la capa de enlace de datos no entienden las direcciones de Internet. En el caso de Ethernet, cada NIC de las que se hayan fabricado viene equipada con una dirección Ethernet única de 48 bits. Los fabricantes de NIC Ethernet solicitan un bloque de direcciones Ethernet al IEEE para asegurar que no haya dos NIC con la misma dirección (y evitar conflictos en caso de que las dos NIC aparezcan alguna vez en la misma LAN). Las NIC envían y reciben tramas basadas en direcciones Ethernet de 48 bits. No saben nada sobre direcciones IP de 32 bits.

La pregunta ahora es: ¿cómo se convierten las direcciones IP en direcciones de la capa de enlace de datos, como Ethernet? Para explicar cómo funciona esto, veamos el ejemplo de la figura 5-61 en donde se muestra una universidad pequeña con dos redes /24. Una red (CC) es una Ethernet conmutada y está en el Departamento de Ciencias Computacionales. Tiene el prefijo 192.32.65.0/24. La otra LAN (IE), que también es Ethernet conmutada, está en el Departamento de Ingeniería Eléctrica y tiene el prefijo 192.32.63.0/24. Las dos LAN están conectadas por un enrutador IP. Cada máquina en una Ethernet y cada interfaz en el enrutador tienen una dirección única de Ethernet, etiquetadas de *E1* a *E6*, además de una dirección IP única en la red CC o IE.

Empecemos por ver cómo un usuario en el host 1 envía un paquete a un usuario en el host 2 de la red CC. Supongamos que el emisor sabe el nombre del receptor pretendido, posiblemente algo así como eagle.cc.uni.edu. El primer paso es encontrar la dirección IP para el host 2. Esta consulta es realizada por



Trama	IP de origen	Eth. de origen	IP de destino	Eth. de destino
Host 1 a 2, en la red CC	IP1	E1	IP2	E2
Host 1 a 4, en la red CC	IP1	E1	IP4	E3
Host 1 a 4, en red IE	IP1	E4	IP4	E6

Figura 5-61. Dos redes LAN Ethernet conmutadas, unidas por un enrutador.

el DNS, que estudiaremos en el capítulo 7. Por el momento supongamos que el DNS devuelve la dirección IP del host 2 (192.32.65.5).

El software de la capa superior en el host 1 elabora ahora un paquete con 192.32.65.5 en el campo *Dirección de destino* y lo entrega al software de IP para que lo transmita. El software IP puede buscar la dirección y ver que el destino está en la red CC (es decir, su propia red). Pero de todas formas necesita alguna manera de encontrar la dirección Ethernet de destino para enviar la trama. Una solución es tener un archivo de configuración en alguna parte del sistema que asocie las direcciones IP con direcciones Ethernet. Aun cuando esta solución es ciertamente posible, para las organizaciones con miles de máquinas, conservar todos estos archivos actualizados es una tarea propensa a errores y consume mucho tiempo.

Una mejor solución es que el host 1 envíe un paquete de difusión hacia Ethernet y pregunte quién posee la dirección IP 192.32.65.5. La difusión llegará a cada máquina en la Ethernet CC y cada una verificará su dirección IP. Al host 2 le bastará responder con su dirección de Ethernet (*E2*). De esta manera, el host 1 aprende que la dirección IP 192.32.65.5 está en el host con la dirección Ethernet *E2*. El protocolo utilizado para hacer esta pregunta y obtener la respuesta se llama **ARP** (**Protocolo de Resolución de Direcciones**, del inglés *Address Resolution Protocol*). Casi todas las máquinas en Internet lo ejecutan. La definición de ARP está en el RFC 826.

La ventaja de usar ARP en lugar de archivos de configuración es su simpleza. El administrador del sistema sólo tiene que asignar a cada máquina una dirección IP y decidir respecto a las máscaras de subred. ARP se hace cargo del resto.

A estas alturas, el software IP en el host 1 crea una trama Ethernet dirigida a *E2*, pone el paquete IP (dirigido a 192.32.65.5) en el campo de carga útil y lo descarga hacia la Ethernet. Las direcciones IP y Ethernet de este paquete se muestran en la figura 5-61. La NIC Ethernet del host 2 detecta esta trama, la reconoce como una trama para sí mismo, la recoge y provoca una interrupción. El controlador de Ethernet extrae el paquete IP de la carga útil y lo pasa al software IP, el cual ve que esté direccionado de forma correcta y lo procesa.

Es posible hacer varias optimizaciones para que ARP trabaje con más eficiencia. Para empezar, una vez que una máquina ha ejecutado ARP, guarda el resultado en caché, en caso de que tenga que ponerse en contacto con la misma máquina en poco tiempo. La siguiente vez encontrará la asociación en su propio caché, con lo cual se elimina la necesidad de una segunda difusión. En muchos casos, el host 2 necesitará devolver una respuesta y se verá forzado también a ejecutar el ARP para determinar la dirección Ethernet del emisor. Podemos evitar esta difusión de ARP haciendo que el host 1 incluya su asociación IP a Ethernet en el paquete ARP. Cuando la difusión de ARP llega al host 2, se introduce el par (192.32.65.7, *E1*) en la caché ARP del host 2. De hecho, todas las máquinas en Ethernet pueden introducir esta asociación en su caché ARP.

Para permitir que las asociaciones cambien, por ejemplo, al configurar un host para que use una nueva dirección IP (pero que mantenga su vieja dirección Ethernet), las entradas en la caché ARP deben expirar después de unos cuantos minutos. Una manera inteligente de ayudar a mantener actualizada la información en la caché y optimizar el desempeño es hacer que cada máquina difunda su asociación cuando se configure. Por lo general, esta difusión se realiza en forma de un ARP que busca su propia dirección IP. No debe haber una respuesta, pero un efecto colateral de la difusión es crear o actualizar una entrada en la caché ARP de todos. A esto se le conoce como **ARP gratuito**. Si una respuesta llega (en forma inesperada), quiere decir que se asignó la misma dirección IP a dos máquinas. El administrador de la red debe resolver este error antes de que ambas máquinas puedan usarla.

Ahora veamos de nuevo la figura 5-61, sólo que esta vez el host 1 quiere enviar un paquete al host 4 (192.32.63.8) en la red IE. El host 1 verá que la dirección IP de destino no está en la red CC. Sabe enviar todo ese tráfico fuera de la red al enrutador, el cual también se conoce como **puerta de enlace predeterminada**. Por convención, la puerta de enlace predeterminada es la dirección más baja en la

red (198.31.65.1). Para enviar una trama al enrutador, el host 1 debe conocer de todas formas la dirección Ethernet de la interfaz del enrutador en la red CC. Para descubrirla envía una difusión ARP para 198.31.65.1, a partir de la cual aprende *E3*. Después envía la trama. Los mismos mecanismos de búsqueda se utilizan para enviar un paquete de un enrutador al siguiente, a través de una secuencia de enrutadores en una ruta de Internet.

Cuando la NIC Ethernet del enrutador recibe esta trama, entrega el paquete al software IP. Sabe con base en las máscaras de red que el paquete se debe enviar a la red IE, en donde alcanzará al host 4. Si el enrutador no conoce la dirección Ethernet para el host 4, entonces usará ARP de nuevo. La tabla en la figura 5-61 lista las direcciones Ethernet e IP de origen y destino que están presentes en las tramas, como se observa en las redes CC e IE. Observe que las direcciones Ethernet cambian con la trama en cada red, mientras que las direcciones IP permanecen constantes (puesto que indican las terminales a través de todas las redes interconectadas).

También es posible enviar un paquete del host 1 al host 4 sin que el primero sepa que el segundo está en una red diferente. La solución es hacer que el enrutador responda a los ARP en la red CC para el host 4 y proporcione su dirección Ethernet, *E3*, como respuesta. No es posible hacer que el host 4 responda directamente, ya que no verá la solicitud ARP (puesto que los enrutadores no reenvían difusiones a nivel de Ethernet). A continuación, el enrutador recibirá las tramas enviadas a 192.32.63.8 y las reenviará a la red IE. A esta solución se le conoce como **ARP por proxy** y se utiliza en casos especiales en los que un host desea aparecer en una red, aun cuando en realidad reside en otra. Por ejemplo, una situación común es una computadora móvil que desea que algún otro nodo recoja los paquetes por ella cuando no se encuentre en su red local.

DHCP: el Protocolo de Configuración Dinámica de Host

ARP (así como los demás protocolos de Internet) asume que los hosts están configurados con cierta información básica, como sus propias direcciones IP. ¿Cómo obtienen los hosts esta información? Es posible configurar en forma manual cada computadora, pero es un proceso tedioso y propenso a errores. Existe una mejor manera de hacerlo, conocida como **DHCP** (**Protocolo de Configuración Dinámica de Host**, del inglés *Dynamic Host Configuration Protocol*).

Con DHCP, cada red debe tener un servidor DHCP responsable de la configuración. Al iniciar una computadora, ésta tiene integrada una dirección Ethernet u otro tipo de dirección de capa de enlace de datos en la NIC, pero no cuenta con una dirección IP. En forma muy parecida al ARP, la computadora difunde una solicitud de una dirección IP en su red. Para ello usa un paquete llamado DHCP DISCOVER. Este paquete debe llegar al servidor DHCP. Si el servidor no está conectado directamente a la red, el enrutador se configurará para recibir difusiones DHCP y transmitirlas al servidor DHCP en donde quiera que se encuentre.

Cuando el servidor recibe la solicitud, asigna una dirección IP libre y la envía al host en un paquete DHCP OFFER (que también se puede transmitir por medio del enrutador). Para que esto pueda funcionar incluso cuando los hosts no tienen direcciones IP, el servidor identifica a un host mediante su dirección Ethernet (la cual se transporta en el paquete DHCP DISCOVER).

Un problema que surge con la asignación automática de direcciones IP de una reserva es determinar qué tanto tiempo se debe asignar una dirección IP. Si un host sale de la red y no devuelve su dirección IP al servidor DHCP, esa dirección se perderá en forma permanente. Después de un tiempo, tal vez se pierdan muchas direcciones. Para evitar que eso ocurra, la asignación de direcciones IP puede ser por un periodo fijo de tiempo, una técnica conocida como **arrendamiento**. Justo antes de que expire el arrendamiento, el host debe pedir una renovación al DHCP. Si no puede hacer una solicitud o si ésta se rechaza, tal vez el host ya no pueda usar la dirección IP que recibió antes.

El DHCP se describe en los RFC 2131 y 2132. Se utiliza ampliamente en Internet para configurar todo tipo de parámetros, además de proporcionar a los hosts direcciones IP. Al igual que en las redes de

negocios y domésticas, los ISP usan DHCP para establecer los parámetros de los dispositivos a través del enlace de acceso a Internet, de modo que los clientes no tengan que comunicarse por teléfono con sus ISP para obtener esta información. Algunos ejemplos comunes de la información que se configura incluyen la máscara de red, la dirección IP de la puerta de enlace predeterminada y las direcciones IP de los servidores DNS y de tiempo. DHCP ha reemplazado en gran parte los protocolos anteriores (conocidos como RARP y BOOTP), con una funcionalidad más limitada.

5.6.5 Conmutación mediante etiquetas y MPLS

Hasta ahora, en nuestro paseo por la capa de red de Internet, nos hemos enfocado exclusivamente en los paquetes como datagramas que los enrutadores IP reenvían. También hay otro tipo de tecnología que está empezando a tener mucho auge, en especial entre los ISP, para desplazar el tráfico de Internet entre sus redes. A esta tecnología se le conoce como **MPLS** (**Conmutación Multiprotocolo Mediante Etiquetas**, del inglés *MultiProtocol Label Switching*) y es peligrosamente cercana a la tecnología de conmutación de circuitos. A pesar del hecho de que muchas personas en la comunidad de Internet tienen una intensa aversión por las redes orientadas a conexión, la idea parece estar regresando. Como Yogi Berra dijo, es como *déjà vu* todo otra vez. Sin embargo, existen diferencias esenciales entre la forma en que Internet maneja la construcción de rutas y la manera en que lo hacen las redes orientadas a conexión, por lo que sin duda la técnica no es una conmutación de circuitos tradicional.

MPLS agrega una etiqueta en frente de cada paquete; el reenvío se basa en la etiqueta en vez de la dirección de destino. Al convertir la etiqueta en un índice de una tabla interna, sólo es cuestión de buscar en la tabla la línea de salida correcta. Mediante el uso de esta técnica, el reenvío se puede llevar a cabo con mucha rapidez. Esta ventaja fue la motivación original detrás de MPLS, que empezó como una tecnología propietaria conocida por varios nombres, incluyendo **conmutación por marcas** (*tag switching*). En un momento dado, la IETF empezó a estandarizar la idea. Se describe en el RFC 3031 y en muchos otros RFC. Con el tiempo, los principales beneficios han sido un enrutamiento flexible y un reenvío adecuado para la calidad del servicio, así como rápido.

La primera pregunta que surge es en dónde colocar la etiqueta. Como los paquetes IP no se diseñaron para circuitos virtuales, no hay un campo disponible para números de circuito virtual dentro del encabezado IP. Por esta razón se tuvo que agregar un nuevo encabezado MPLS enfrente del encabezado IP. En una línea de enrutador a enrutador que utiliza PPP como protocolo de entramado, el formato de trama, incluyendo los encabezados PPP, MPLS, IP y TCP, es como se muestra en la figura 5-62.

El encabezado MPLS genérico tiene una longitud de 4 bytes y cuatro campos. El más importante es el campo *Etiqueta* que contiene el índice. El campo *QoS* indica la clase de servicio. El campo *S* se relaciona con el apilamiento de múltiples etiquetas (lo que veremos a continuación). El campo *TtL* indica cuántas

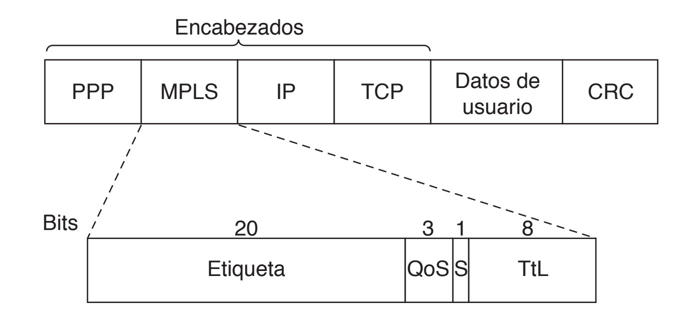


Figura 5-62. Transmisión de un segmento TCP mediante IP, MPLS y PPP.

veces se puede reenviar el paquete. Se decrementa en cada enrutador y, si llega a 0, se descarta el paquete. Esta característica evita los ciclos infinitos en caso de inestabilidad del enrutamiento.

MPLS se encuentra entre el protocolo IP de la capa de red y el protocolo PPP de la capa de enlace. En realidad no es un protocolo de capa 3, pues depende de direcciones IP o de otra capa de red para establecer las rutas de las etiquetas. En realidad tampoco es un protocolo de capa 2, pues reenvía los paquetes a través de varios saltos, no de un solo enlace. Por esta razón, algunas veces se denomina protocolo de capa 2.5. Es una ilustración de que los protocolos reales no siempre se ajustan perfectamente a nuestro modelo ideal de protocolos por capas.

Por el lado optimista, como los encabezados MPLS no son parte del paquete de capa de red ni de la trama de capa de datos, MPLS es en gran parte independiente de ambas capas. Entre otras cosas, esta propiedad significa que es posible construir switches MPLS que puedan reenviar tanto paquetes IP como paquetes que no sean IP, dependiendo de lo que se presente. Esta característica es la razón del término "multiprotocolo" en el nombre de MPLS. También puede transportar paquetes IP sobre redes que no sean IP.

Cuando llega un paquete con capacidad para MPLS a un **LSR** (**Enrutador de Conmutación de Etiquetas**, del inglés *Label Switched Router*), la etiqueta se utiliza como un índice en una tabla para determinar la línea de salida y la nueva etiqueta a utilizar. Esta conmutación de etiquetas se utiliza en todas las redes de circuitos virtuales. Las etiquetas sólo tienen importancia local y dos enrutadores diferentes pueden transmitir paquetes no relacionados que contengan la misma etiqueta hacia otro enrutador para transmitirlos en la misma línea de salida. Para diferenciarlos en el otro extremo, las etiquetas se tienen que volver a asociar en cada salto. En la figura 5-3 vimos este mecanismo en acción. MPLS utiliza la misma técnica.

Como observación adicional, algunas personas hacen la distinción entre *reenvío* y *conmutación*. El reenvío es el proceso de encontrar la mejor coincidencia para una dirección de destino en una tabla y decidir por dónde enviar los paquetes. Un ejemplo es el algoritmo del prefijo más largo coincidente que se utiliza para el reenvío IP. En contraste, la conmutación usa una etiqueta que se toma del paquete como un índice hacia una tabla de reenvío. Es más simple y más rápido. Sin embargo, estas definiciones están lejos de ser universales.

Como la mayoría de los hosts y enrutadores no entienden MPLS, también deberíamos preguntar cómo y cuándo se adjuntan las etiquetas a los paquetes. Esto ocurre cuando un paquete IP llega al extremo de una red MPLS. El **LER** (**Enrutador de Etiquetas de Borde**, del inglés *Label Edge Router*) inspecciona la dirección IP de destino y otros campos para ver qué ruta MPLS debe seguir el paquete, y coloca la etiqueta correcta al frente del paquete. Dentro de la red MPLS, esta etiqueta se utiliza para reenviar el paquete. En el otro extremo de la red MPLS, la etiqueta ha cumplido su propósito y se elimina para revelar el paquete IP a la siguiente red. Este proceso se muestra en la figura 5-63. Una diferencia de los circuitos virtuales tradicionales es el nivel de agregación. Sin duda es posible que cada flujo tenga su propio conjunto de etiquetas a través de la red MPLS. Sin embargo, es más común que los enrutadores agrupen varios flujos que terminan en un enrutador o LAN en particular, y que usen una sola etiqueta para ellos.

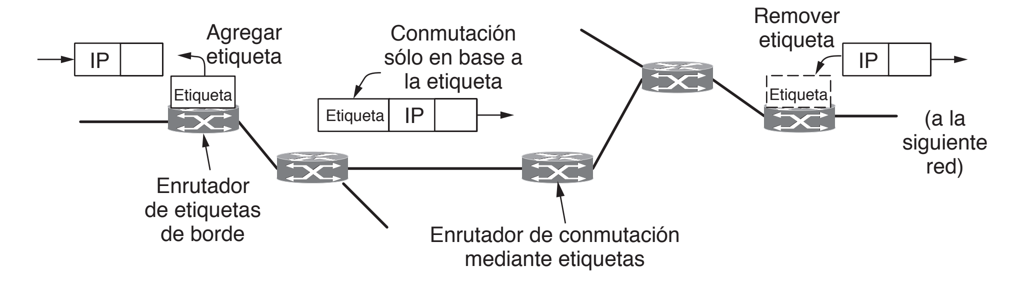


Figura 5-63. Reenvío de un paquete IP por medio de una red MPLS.

Se dice que los flujos que se agrupan bajo una sola etiqueta pertenecen a la misma **FEC** (**Clase de Equivalencia de Reenvío**, del inglés *Forwarding Equivalence Class*). Esta clase no sólo cubre hacia dónde van los paquetes, sino también su clase de servicio (en el sentido de los servicios diferenciados), ya que todos los paquetes se tratan de la misma forma para fines de reenvío.

En el enrutamiento tradicional de circuitos virtuales, no es posible agrupar varias rutas distintas con diferentes terminales en el mismo identificador de circuito virtual, ya que no habría forma de diferenciarlas en el destino final. Con MPLS, los paquetes aún contienen su dirección de destino final, además de la etiqueta. Al final de la ruta etiquetada, se puede eliminar el encabezado de etiqueta y el reenvío puede proseguir de la forma usual, mediante el uso de la dirección de destino de la capa de red.

En realidad, MPLS va aún más allá. Puede operar a múltiples niveles al mismo tiempo, para lo cual agrega más de una etiqueta al frente de un paquete. Por ejemplo, suponga que hay muchos paquetes que ya tienen distintas etiquetas (puesto que deseamos tratar a los paquetes de manera distinta en alguna parte de la red), las cuales deben seguir una ruta común hacia cierto destino. En vez de establecer muchas rutas de conmutación, una para cada una de las distintas etiquetas, podemos establecer una sola ruta. Cuando los paquetes ya etiquetados llegan al inicio de esta ruta, se agrega otra etiqueta al frente. A esto se le conoce como pila de etiquetas. La etiqueta más externa guía a los paquetes a lo largo de la ruta. La etiqueta se elimina al final de la ruta y las etiquetas que se revelan, si las hay, se utilizan para reenviar el paquete el resto del camino. El bit *S* en la figura 5-62 permite a un enrutador eliminar una etiqueta para saber si quedan etiquetas adicionales. Se establece en 1 para la etiqueta inferior y en 0 para todas las demás etiquetas.

La pregunta final por hacer es cómo se establecen las tablas de reenvío de etiquetas de manera que los paquetes las sigan. Ésta es un área de gran diferencia entre los diseños de MPLS y de los circuitos virtuales convencionales. En las redes tradicionales de circuitos virtuales, cuando un usuario desea establecer una conexión, se lanza un paquete de establecimiento por la red para crear la ruta y las entradas en la tabla de reenvío. MPLS no involucra a los usuarios en la fase de establecimiento. Exigir a los usuarios que hagan algo además de enviar un datagrama quebrantaría a la mayor parte del software existente en Internet.

En cambio, la información de reenvío se establece mediante protocolos que son una combinación de protocolos de enrutamiento y de establecimiento de conexión. Estos protocolos de control están totalmente separados del reenvío de etiquetas, lo cual nos permite usar múltiples protocolos de control distintos. Una de las variantes funciona así. Cuando se enciende un enrutador, éste revisa cuáles son las rutas para las que será el destino final (por ejemplo, qué prefijos pertenecen a sus interfaces). Después crea una o más FECs para estas rutas, asigna una etiqueta para cada una de ellas y pasa las etiquetas a sus vecinos. Éstos a su vez, introducen las etiquetas en sus tablas de reenvío y envían nuevas etiquetas a sus vecinos, hasta que todos los enrutadores hayan adquirido la ruta. También se pueden reservar recursos a medida que se construye la ruta para garantizar una calidad de servicio apropiada. Otras variantes pueden establecer distintas rutas, como las de ingeniería de tráfico que toman en cuenta la capacidad no utilizada, y crean rutas bajo demanda para soportar ofrecimientos de servicios tales como la calidad del servicio.

Aunque las ideas básicas detrás de MPLS son simples y directas, los detalles son complicados, con muchas variaciones y casos de uso en desarrollo constante. Para obtener más información, consulte a Davie y Farrel (2008), y también a Davie y Rekhter (2000).

5.6.6 OSPF: un protocolo de enrutamiento de puerta de enlace interior

Hemos terminado nuestro estudio acerca de cómo se reenvían los paquetes en Internet. Ya es tiempo para pasar al tema siguiente: el enrutamiento en Internet. Como lo mencionamos antes, Internet se compone

de una gran cantidad de redes independientes o **Sistemas Autónomos** (**AS**), los cuales son operados por distintas organizaciones, por lo general una empresa, universidad o ISP. Dentro de su propia red, una organización puede usar su propio algoritmo de enrutamiento interno, o **enrutamiento intradominio**, como se le conoce con más frecuencia. Sin embargo, hay sólo unos cuantos protocolos estándar que son populares. En esta sección estudiaremos el problema del enrutamiento intradominio y analizaremos el protocolo OSPF, que se utiliza mucho en la práctica. Un protocolo de enrutamiento intradominio también se conoce como **protocolo de puerta de enlace interior**. En la siguiente sección, estudiaremos el problema de enrutamiento entre redes operadas de manera independiente, o **enrutamiento interdominio**. Para ese caso, todas las redes deben usar el mismo protocolo de enrutamiento interdominio, o **protocolo de puerta de enlace exterior**. El protocolo que se utiliza en internet es **BGP** (**Protocolo de Puerta de Enlace de Frontera**, del inglés *Border Gateway Protocol*).

Los primeros protocolos de enrutamiento intradominio usaban un diseño de vector de distancia basado en el algoritmo Bellman-Ford distribuido, que se heredó de ARPANET. RIP (Protocolo de Información de Enrutamiento, del inglés *Routing Information Protocol*) es el principal ejemplo que se usa en la actualidad. Funciona bien en sistemas pequeños, pero su desempeño disminuye a medida que las redes aumentan su tamaño. También sufre del problema del conteo al infinito y por lo general de una convergencia lenta. ARPANET cambió a un protocolo de estado del enlace en mayo de 1979 debido a estos problemas; en 1988, la IETF empezó a trabajar en un protocolo de estado del enlace para el enrutamiento intradominio. Ese protocolo, conocido como **OSPF** (**Abrir primero la ruta más corta**, del inglés *Open Shortest Path First*), se convirtió en un estándar en 1990. Se valió de un protocolo llamado **IS-IS** (**Sistema Intermedio a Sistema Intermedio**, del inglés *Intermediate-System to Intermediate-System*), el cual se convirtió en un estándar de ISO. Debido a su herencia compartida, los dos protocolos son mucho más parecidos que diferentes. Si desea conocer la historia completa, consulte el RFC 2328. Son los protocolos de enrutamiento intradominio dominantes; la mayoría de los distribuidores ofrecen ahora soporte para ambos. OSPF se utiliza más en las redes de compañías; IS-IS se utiliza más en las redes de ISP. De los dos, veremos un bosquejo sobre la forma en que funciona OSPF.

Dada la extensa experiencia con otros protocolos de enrutamiento, el grupo diseñador de OSPF tenía una larga lista de requerimientos por cumplir. Primero, había que publicar el algoritmo en la literatura abierta, de aquí que se incluya una "O" en OSPF. No sería factible usar una solución propietaria perteneciente a una sola compañía. Segundo, el nuevo protocolo tenía que soportar una variedad de métricas de distancia, incluyendo la distancia física, el retardo, etc. Tercero, tenía que ser un algoritmo dinámico, uno que se adaptara a los cambios en la topología de manera automática y rápida.

Cuarto (y lo que constituía una novedad para OSPF), tenía que soportar un enrutamiento con base en el tipo de servicio. El nuevo protocolo tenía que ser capaz de enrutar el tráfico en tiempo real de una manera y el tráfico restante de una manera distinta. En ese tiempo, IP tenía un campo *Tipo de servicio* pero ningún protocolo existente lo usaba. Este campo se incluyó en OSPF pero de todas formas nadie lo usó, por lo que se eliminó eventualmente. Tal vez este requerimiento estaba adelantado a su tiempo, pues precedió al trabajo de la IETF sobre los servicios diferenciados, lo cual ha rejuvenecido las clases de servicio.

Quinto (y con relación a lo anterior), OSPF tenía que balancear las carga para dividirla entre múltiples líneas. Los protocolos anteriores enviaban todos los paquetes a través de una mejor ruta, incluso si había dos rutas igual de buenas. La otra ruta no se utilizaba para nada. En muchos casos, dividir la carga entre múltiples rutas produce un mejor desempeño.

Sexto, se necesitaba un soporte para los sistemas jerárquicos. Para 1988 algunas redes habían crecido tanto que no se podía esperar que un enrutador conociera toda la topología. Había que diseñar a OSPF de modo que ningún enrutador tuviera que hacerlo.

Séptimo, se requería un mínimo de seguridad para evitar que los estudiantes en busca de diversión se burlaran de los enrutadores al enviarles información de enrutamiento falsa. Por último, había que tomar

las medidas necesarias para lidiar con los enrutadores conectados a Internet por medio de un túnel. Los protocolos anteriores no manejaban bien esto.

OSPF soporta los enlaces punto a punto (por ejemplo, SONET) y las redes de difusión (la mayoría de las LAN). En realidad, es capaz de soportar redes con múltipes enrutadores, cada uno de los cuales se puede comunicar en forma directa con los demás (a éstas se les conoce como **redes multiacceso**) incluso aunque no tengan capacidad de difusión. Los primeros protocolos no manejaban bien este caso.

En la figura 5-64(a) se muestra un ejemplo de una red con un sistema autónomo. Se omiten los hosts debido a que en general no desempeñan ningún papel en OSPF, mientras que los enrutadores y las redes (que pueden contener hosts) sí. La mayoría de los enrutadores en la figura 5-64(a) están conectados a otros enrutadores mediante enlaces punto a punto, y a redes para llegar a los hosts en ellas. Sin embargo, los enrutadores *R3*, *R4* y *R5* están conectados mediante una LAN de difusión tal como una red Ethernet conmutada.

Para operar, OSPF resume la colección de redes reales, enrutadores y enlaces en un grafo dirigido en el que a cada arco se le asigna un peso (distancia, retardo, etc.). Una conexión punto a punto entre dos enrutadores se representa por un par de arcos, uno en cada dirección. Sus pesos pueden ser diferentes. Una red de difusión se representa con un nodo para la red en sí, más un nodo para cada enrutador. Los arcos de ese nodo de la red a los enrutadores tienen un peso de 0. Sin embargo son importantes, puesto que sin ellos no habrá una ruta a través de la red. Otras redes, que sólo tienen hosts, tienen únicamente un arco que llega a ellas y no uno que regresa. Esta estructura proporciona rutas a los hosts, pero no a través de ellos.

La figura 5-64(b) muestra la representación gráfica de la red de la figura 5-64(a). En esencia, lo que OSPF hace es representar la red real mediante un grafo como éste y después usa el método de estado del enlace para hacer que cada enrutador calcule la ruta más corta desde sí mismo hacia todos los demás nodos. Se pueden encontrar varias rutas que sean igual de cortas. En este caso, OSPF recuerda el conjunto de rutas más cortas y, durante el reenvío de paquetes, el tráfico se divide entre ellas. Este proceso ayuda a balancear la carga y se conoce como **ECMP** (**Multiruta de Igual Costo**, del inglés *Equal Cost MultiPath*).

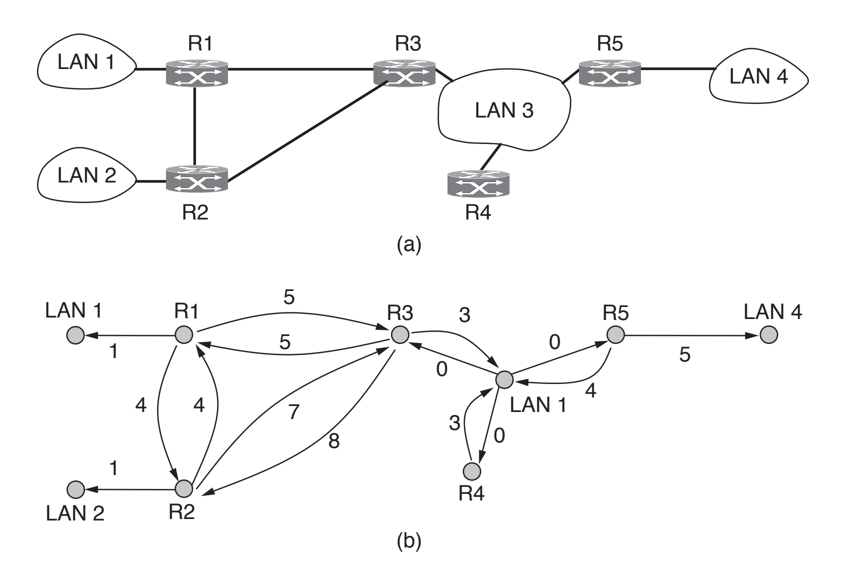


Figura 5-64. (a) Un sistema autónomo. (b) Una representación gráfica de (a).

Muchos de los sistemas autónomos (AS) en Internet son grandes por sí mismos y nada sencillos de administrar. Para trabajar a esta escala, OSPF permite dividir un AS en **áreas** numeradas, en donde un área es una red o un conjunto de redes contiguas. Las áreas no se traslapan ni necesitan ser exhaustivas; es decir, algunos enrutadores no necesitan pertenecer a ningún área. Los enrutadores que están totalmente dentro de un área se llaman **enrutadores internos**. Un área es una generalización de una red individual. Fuera de un área, sus destinos son visibles pero su topología no. Esta característica ayuda a escalar el enrutamiento.

Cada AS tiene un **área troncal** (*backbone area*), llamada área 0. Los enrutadores en esta área se llaman **enrutadores troncales** (*backbone routers*). Todas las áreas se conectan a la red troncal, posiblemente mediante túneles, de modo que es posible ir desde cualquier área en el AS a cualquier otra área en el AS mediante la red troncal. En el grafo, un túnel se representa como otro arco más con un costo. Al igual que con otras áreas, la topología de la red troncal no es visible fuera de ésta.

Cada enrutador que se conecta a dos o más áreas se denomina **enrutador de frontera de área**. También debe formar parte de la red troncal. El trabajo de un enrutador de frontera de área es resumir todos los destinos en un área e inyectar este resumen en las otras áreas a las que está conectado. El resumen incluye la información de costo, pero no todos los detalles de la topología dentro de un área. Pasar la información de costo, permite a los hosts en otras áreas encontrar el mejor enrutador de frontera de área que puede usar para entrar a un área. Al no pasar la información sobre la topología se reduce el tráfico y se simplifican los cálculos de la ruta más corta para los enrutadores en otras áreas. No obstante, si sólo hay un enrutador de frontera fuera de un área, no es necesario ni siquiera pasar el resumen. Los enrutadores a destinos fuera del área siempre empiezan con la instrucción “Ir al enrutador de frontera”. Este tipo de área se denomina **área aislada** (*stub area*).

El último tipo de enrutador es el **enrutador de límite de AS**. Éste inyecta en el área las rutas a destinos externos en otros sistemas autónomos. Después, las rutas externas aparecen como destinos a los que se puede llegar por medio del enrutador de límite de AS con un cierto costo. Una ruta externa se puede inyectar en uno o más enrutadores de límite de AS. En la figura 5-65 se muestra la relación entre los sistemas autónomos, las áreas y los diversos tipos de enrutadores. Un enrutador puede desempeñar distintos roles; por ejemplo, un enrutador de frontera puede ser también un enrutador troncal.

Durante la operación normal, cada enrutador dentro de un área tiene la misma base de datos de estado del enlace y ejecuta el mismo algoritmo de la ruta más corta. Su tarea principal es calcular la ruta más corta desde sí mismo hasta cada uno de los demás enrutadores y redes en todo el AS. Un enrutador de frontera de área necesita las bases de datos para todas las áreas a las que está conectado y debe ejecutar el algoritmo de la ruta más corta para cada área por separado.

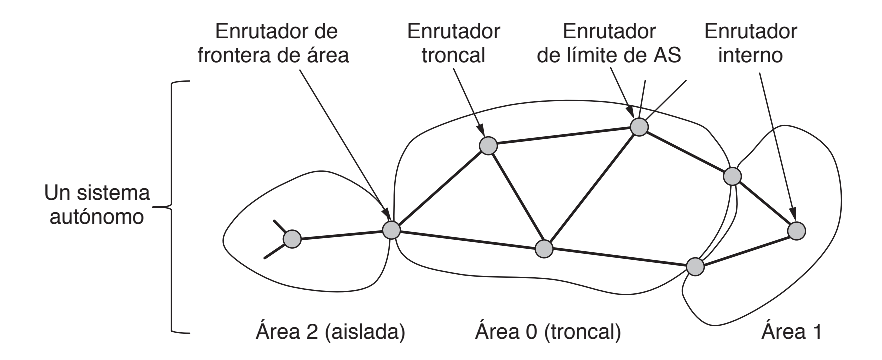


Figura 5-65. La relación entre sistemas autónomos, redes troncales y áreas en OSPF.

Para un origen y un destino en la misma área, se selecciona la mejor ruta intra-área (que se encuentre totalmente dentro del área). Para un origen y un destino en distintas áreas, la ruta interárea debe ir del origen a la red troncal, a través de la red troncal hasta el área de destino, y después al destino. Este algoritmo obliga a una configuración de estrella en OSPF, en donde la red troncal es el hub y las demás áreas son las extensiones. Puesto que se selecciona la ruta con el menor costo, es probable que los enrutadores en distintas partes de la red usen distintos enrutadores de frontera de área para entrar a la red troncal y al área de destino. Los paquetes se enrutan del origen al destino "como están". No se encapsulan ni tunelizan (a menos que vayan a un área cuya única conexión a la red troncal sea un túnel). Además, las rutas a los destinos externos pueden incluir el costo externo del enrutador de límite de AS a través de la ruta externa, si se desea, o sólo el costo interno para el AS.

Cuando un enrutador se enciende, envía mensajes HELLO en todas sus líneas punto a punto y los envía por multidifusión a las LAN al grupo que consiste en los enrutadores restantes. Cada enrutador descubre quiénes son sus vecinos gracias a las respuestas. Todos los enrutadores en la misma LAN son vecinos.Para operar, OSPF intercambia información entre enrutadores adyacentes, que no es lo mismo que entre enrutadores vecinos. En particular, es ineficiente que cada enrutador en la LAN se comunique con cualquier otro enrutador en la LAN. Para evitar esta situación, se elige un enrutador como **enrutador designado**. Se dice que es **adyacente** a todos los demás enrutadores en su LAN, e intercambia información con ellos. En efecto, actúa como el único nodo que representa a la LAN. Los enrutadores vecinos que no son adyacentes no intercambian información entre sí. Un enrutador designado como respaldo siempre se mantiene actualizado para facilitar la transición en caso de que el enrutador designado primario falle y haya que reemplazarlo de inmediato.

Durante la operación normal, cada enrutador inunda periódicamente con mensajes *LINK STATE UPDATE* (ACTUALIZACIÓN DEL ESTADO DEL ENLACE) a cada uno de sus enrutadores adyacentes. Estos mensajes dan su estado y proporcionan los costos usados en la base de datos topológica. Para hacerlos confiables, se confirma la recepción de los mensajes de inundación. Cada mensaje tiene un número de secuencia para que un enrutador pueda ver si un *LINK STATE UPDATE* entrante es más viejo o más nuevo que el que tiene actualmente. Los enrutadores también envían estos mensajes cuando un enlace se activa o desactiva, o cuando cambia su costo.

Los mensajes *DATABASE DESCRIPTION* (DESCRIPCIÓN DE LA BASE DE DATOS) dan los números de secuencia de todas las entradas de estado del enlace que contiene el emisor en ese momento. Al comparar sus propios valores con los del emisor, el receptor puede determinar quién tiene los valores más recientes. Estos mensajes se usan cuando se activa un enlace.

Cualquier socio puede solicitar información del estado del enlace al otro mediante los mensajes *LINK STATE REQUEST* (PETICIÓN DEL ESTADO DEL ENLACE). El resultado de este algoritmo es que cada par de enrutadores adyacentes verifica quién tiene los datos más recientes; de esta manera se difunde la nueva información a lo largo del área. Todos estos mensajes se envían de manera directa como paquetes IP. En la figura 5-66 se resumen los cinco tipos de mensajes.

Tipo de mensaje	Descripción
Hello.	Se utiliza para descubrir quiénes son los vecinos.
Link state update.	Proporciona los costos del emisor a sus vecinos.
Link state ack.	Confirma la recepción de la actualización del estado del enlace.
Database description.	Anuncia qué actualizaciones tiene el emisor.
Link state request.	Solicita información del socio.

Figura 5-66. Los cinco tipos de mensajes OSPF.

Finalmente podemos reunir todas las piezas. Mediante la inundación de mensajes, cada enrutador informa a todos los demás enrutadores en su área sobre sus enlaces a otros enrutadores y redes, junto con el costo de éstos. Esta información permite a cada enrutador construir el grafo para su(s) área(s) y calcular las rutas más cortas. El área red troncal también hace esto. Además, los enrutadores troncales aceptan la información de los enrutadores de frontera de área para calcular la mejor ruta desde cada enrutador troncal hasta cada uno de los demás enrutadores. Esta información se propaga de vuelta a los enrutadores de frontera de área, quienes la anuncian dentro de sus áreas. Mediante el uso de esta información, los enrutadores internos pueden seleccionar la mejor ruta a un destino fuera de su área, incluyendo el mejor enrutador de salida hacia la red troncal.

5.6.7 BGP: el protocolo de enrutamiento de Puerta de Enlace Exterior

Dentro de un solo sistema autónomo, OSPF e IS-IS son los protocolos de uso común. Entre los sistemas autónomos se utiliza un protocolo diferente, conocido como **BGP** (**Protocolo de Puerta de Enlace de Frontera**, del inglés *Border Gateway Protocol*). Se necesita un protocolo diferente debido a que los objetivos de un protocolo intradominio y de un protocolo interdominio no son los mismos. Todo lo que tiene que hacer un protocolo intradominio es mover paquetes de la manera más eficiente posible desde el origen hasta el destino. No tiene que preocuparse por las políticas.

En contraste, los protocolos de enrutamiento interdominio tienen que preocuparse en gran manera por la política (Metz, 2001). Por ejemplo, tal vez un sistema autónomo corporativo desee la habilidad de enviar paquetes a cualquier sitio de Internet y recibir paquetes de cualquier sitio de Internet. Sin embargo, quizás no esté dispuesto a llevar paquetes de tránsito que se originen en un AS foráneo y estén destinados a un AS foráneo diferente, aun cuando su propio AS se encuentre en la ruta más corta entre los dos sistemas autónomos foráneos ("Ése es su problema, no el nuestro"). Por otro lado, podría estar dispuesto a llevar el tráfico del tránsito para sus vecinos o incluso para otros sistemas autónomos específicos que hayan pagado por este servicio. Por ejemplo, las compañías telefónicas podrían estar contentas de actuar como empresas portadoras para sus clientes, pero no para otros. En general, los protocolos de puerta de enlace exterior (y BGP en particular) se han diseñado para permitir que se implementen muchos tipos de políticas de enrutamiento en el tráfico entre sistemas autónomos.

Las políticas típicas implican consideraciones políticas, de seguridad, o económicas. Algunos ejemplos de posibles restricciones de enrutamiento son:

1. No transportar tráfico comercial en la red educativa.
2. Nunca enviar tráfico del Pentágono por una ruta a través de Irak.
3. Usar TeliaSonera en vez de Verizon porque es más económico.
4. No usar AT&T en Australia porque el desempeño es pobre.
5. El tráfico que empieza o termina en Apple no debe transitar por Google.

Como puede imaginar de esta lista, las políticas de enrutamiento pueden ser muy individuales. A menudo son propietarias pues contienen información comercial delicada. Sin embargo, podemos describir algunos patrones que capturan el razonamiento anterior de la compañía y que se utilizan con frecuencia como un punto de partida.

Para implementar una política de enrutamiento hay que decidir qué tráfico puede fluir a través de cuáles enlaces entre los sistemas autónomos. Una política común es que un ISP cliente pague a otro ISP proveedor por entregar paquetes a cualquier otro destino en Internet y recibir los paquetes enviados desde cualquier otro destino. Se dice que el ISP cliente compra **servicio de tránsito** al ISP proveedor. Es justo igual que cuando un cliente doméstico compra servicio de acceso a Internet con un ISP. Para que funcione,

el proveedor debe anunciar las rutas a todos los destinos en Internet al cliente a través del enlace que los conecta. De esta forma, el cliente tendrá una ruta para enviar paquetes a cualquier parte. Por el contrario, el cliente sólo debe anunciar al proveedor las rutas a los destinos en su red. Esto permitirá al proveedor enviar tráfico al cliente sólo para esas direcciones; al cliente no le conviene manejar el tráfico destinado a otras partes.

En la figura 5-67 podemos ver un ejemplo del servicio de tránsito. Hay cuatro sistemas autónomos conectados. Con frecuencia, la conexión se hace mediante un enlace en **IXPs** (**Puntos de Intercambio de Internet**, del inglés *Internet eXchange Points*): instalaciones con las que muchos ISP tienen un enlace para fines de conectarse con otros ISP. *AS2*, *AS3* y *AS4* son clientes de *AS1*, pues le compran servicio de tránsito. Así, cuando la fuente *A* envía al destino *C*, los paquetes viajan de *AS2* hacia *AS1* y finalmente a *AS4*. Los anuncios de enrutamiento viajan en dirección opuesta a los paquetes. *AS4* anuncia a *C* como un destino a su proveedor de tránsito *AS1*, para que las fuentes puedan llegar a *C* por medio de *AS1*. Después, *AS1* anuncia a sus otros clientes, incluyendo *AS2*, una ruta a *C* para que éstos sepan que pueden enviar tráfico a *C* por medio de *AS1*.

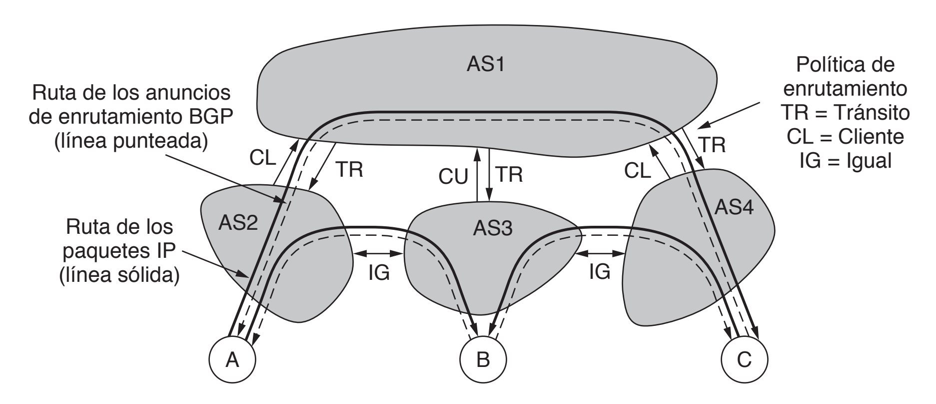


Figura 5-67. Políticas de enrutamiento entre cuatro sistemas autónomos.

En la figura 5-67, los demás sistemas autónomos compran servicio de tránsito a *AS1*. Este servicio les proporciona conectividad para que puedan interactuar con cualquier host en Internet. Sin embargo, tienen que pagar por este privilegio. Suponga que *AS2* y *AS3* intercambian mucho tráfico. Dado que sus redes ya se encuentran conectadas, si lo desean pueden usar una política diferente: pueden enviar tráfico directamente uno al otro sin costo. Esto reducirá la cantidad de tráfico que debe entregar *AS1* a cuenta de *AS2 y AS3*, y con suerte reducirá sus facturas. A esta política se le conoce como **comunicación entre pares** (*peering*).

Para implementar la comunicación entre pares, dos sistemas autónomos se envían anuncios de enrutamiento entre sí, respecto a las direcciones que residen en sus redes. Al hacer esto, *AS2* puede enviar a *AS3* paquetes de *A* destinados a *B* y viceversa. Sin embargo, hay que tener en cuenta que la comunicación entre iguales no es transitiva. En la figura 5-67, *AS3* y *AS4* también se comunican entre sí. Esta comunicación de igual a igual permite que el tráfico de *C,* que está destinado a *B,* se envíe directamente a *AS4*. ¿Qué ocurre si *C* envía un paquete a *A*? *AS3* sólo está anunciando a *AS4* una ruta a *B*. No está anunciando una ruta a *A*. La consecuencia es que el tráfico no pasará de *AS4* a *AS3* a *AS2*, aun cuando existe una ruta física. Esta restricción es justo lo que *AS3* quiere. Se comunica con *AS4* para intercambiar tráfico, pero no quiere transportar tráfico de *AS4* a otras partes de Internet, ya que no se le paga por hacerlo. En cambio, *AS4* recibe servicio de tránsito de *AS1*. Por ende, es *AS1*quien transportará el paquete de *C* a *A*.

Ahora que sabemos sobre el tránsito y la comunicación entre iguales, también podemos ver que *A*, *B* y *C* tienen arreglos de tránsito. Por ejemplo, *A* debe comprar acceso a Internet a *AS2*. *A* podría ser una sola computadora doméstica o la red de una compañía con muchas LAN. Sin embargo, no necesita ejecutar BGP debido a que es una **red aislada** (*stub network*) que está conectada al resto de Internet sólo mediante un enlace. Por tanto, el único lugar para enviar paquetes destinados a puntos que estén fuera de la red es a través del enlace a *AS2*. No hay ningún otro lugar a dónde ir. Para arreglar esta ruta, sólo hay que establecer una ruta predeterminada. Por esta razón no hemos mostrado a *A*, *B* y *C* como sistemas autónomos que participan en el enrutamiento interdominio.

Por otro lado, las redes de algunas compañías están conectadas a varios ISP. Esta técnica se utiliza para mejorar la confiabilidad, ya que si la ruta a través de un ISP falla, la compañía puede usar la ruta a través del otro ISP. Esta técnica se conoce como **multihoming**. En este caso, la red de la compañía probablemente ejecute un protocolo de enrutamiento interdominio (como BGP) para indicar a otros sistemas autónomos qué enlaces de ISP pueden llegar a cuáles direcciones.

Hay muchas variaciones posibles de estas políticas de tránsito y comunicación entre iguales, pero todas ilustran cómo las relaciones de negocios y el control sobre el camino que pueden tomar los anuncios de rutas pueden implementar distintos tipos de políticas. Ahora consideraremos con más detalle cómo los enrutadores que ejecutan BGP se anuncian rutas entre sí y seleccionan rutas a través de las cuales pueden reenviar los paquetes.

BGP es una forma de protocolo de vector de distancia, aunque es bastante distinto a los protocolos de vector de distancia intradominio como RIP. Ya vimos antes que la política, y no la distancia mínima, se utiliza para elegir qué rutas usar. Otra gran diferencia es que, en vez de mantener sólo el costo de la ruta a cada destino, cada enrutador BGP lleva el registro de la ruta utilizada. Esta metodología se conoce como **protocolo de vector de ruta**. La ruta consiste en el enrutador del siguiente salto (que puede estar del otro lado del ISP y no necesariamente ser adyacente) y la secuencia de sistemas autónomos (o **ruta AS**) en el recorrido (se proporciona en orden inverso). Por último, los pares de enrutadores BGP se comunican entre sí mediante el establecimiento de conexiones TCP. Al operar de esta forma se obtiene una comunicación confiable; además se ocultan todos los detalles de la red que se está atravesando.

En la figura 5-68 se muestra un ejemplo de cómo se anuncian las rutas BGP. Hay tres sistemas autónomos y el de en medio provee tránsito a los ISP izquierdo y derecho. Un anuncio de ruta para el prefijo *C* empieza en *AS3*. Cuando se propaga a través del enlace a *R2c* en la parte superior de la figura, tiene la ruta AS que consiste tan sólo en *AS3* y el enrutador del siguiente salto de *R3a*. En la parte inferior tiene la misma ruta AS pero un siguiente salto distinto, debido a que provino de un enlace diferente. Este anuncio continúa su propagación y atraviesa el límite hacia *AS1*. En el enrutador *R1a*, en la parte superior de la figura, la ruta AS es *AS2*, *AS3* y el siguiente salto es *R2a*.

Al transportar la ruta completa con el recorrido, es fácil para el enrutador receptor detectar e interrumpir los ciclos de enrutamiento. La regla es que cada enrutador que envíe un recorrido hacia fuera del AS anteponga su propio número de AS a este recorrido (ésta es la razón por la cual la lista está en orden inverso). Cuando un enrutador recibe un recorrido, verifica que su propio número de AS ya se encuentre en la ruta AS. Si es así, se ha detectado un ciclo y se descarta el anuncio. No obstante (aunque suene un poco irónico), a finales de la década de 1990 se descubrió que a pesar de esta precaución, BGP sufre de una versión del problema de conteo al infinito (Labovitz y colaboradores, 2011). No hay ciclos de larga duración, pero algunas veces los recorridos pueden ser lentos para converger y tienen ciclos transitorios.

Proporcionar una lista de sistemas autónomos es una forma muy burda de especificar una ruta. Un AS podría ser una compañía pequeña o una red troncal internacional. No hay forma de saberlo con base en la ruta. BGP ni siquiera lo intenta, ya que los distintos sistemas autónomos pueden usar protocolos intradominio diferentes, cuyos costos no se puedan comparar. Incluso si se pudieran comparar, tal vez un AS

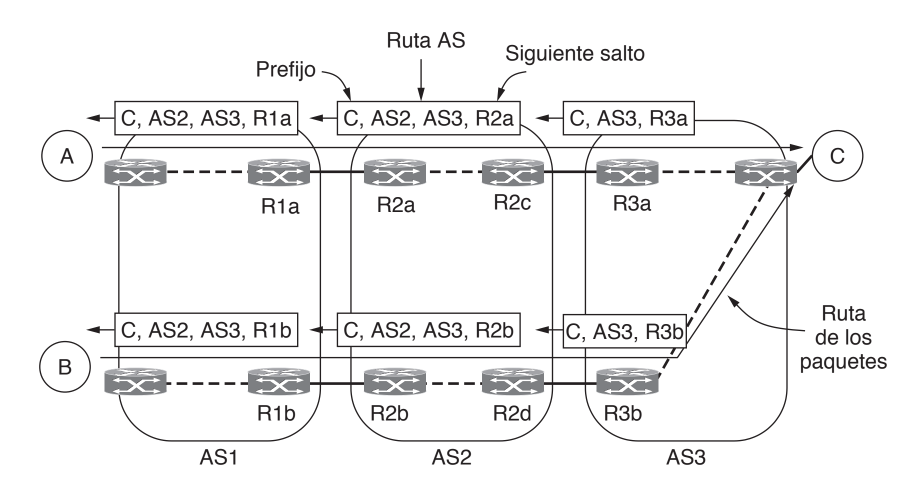


Figura 5-68. Propagación de los anuncios de rutas de BGP.

no quiera revelar su métrica interna. Ésta es una de las diferencias entre los protocolos de enrutamiento interdominio y los protocolos intradominio.

Hasta ahora hemos visto cómo se envía un anuncio de ruta a través del enlace entre dos ISP. Todavía se necesiata una manera de propagar rutas BGP de un lado del ISP al otro, para que se puedan enviar al siguiente ISP. El protocolo intradominio podría manejar esta tarea, pero ya que BGP es muy bueno para escalar a grandes redes, lo común es utilizar una variante de este protocolo, conocido como **iBGP** (**BGP interno**, del inglés *internal BGP*) para diferenciarlo del uso regular de BGP como **eBGP** (**BGP externo**, del inglés *external BGP*).

La regla para propagar rutas dentro de un ISP es que cada enrutador en el límite del ISP aprenda de todas las rutas vistas por todos los demás enrutadores de límite, para fines de consistencia. Si un enrutador de límite en el ISP aprende un prefijo para el IP 128.208.0.0/16, todos los demás enrutadores conocerán este prefijo. Así, se podrá llegar al prefijo desde cualquier parte del ISP, sin importar cuántos paquetes entren al ISP provenientes de otros sistemas autónomos.

No mostramos esta propagación en la figura 5-68 para evitar un desorden, pero por ejemplo, el enrutador *R2b* sabrá que puede llegar a *C* por medio del enrutador *R2c* en la parte superior o del enrutador *R2d* en la parte inferior. El siguiente salto se actualiza a medida que la ruta atraviesa el interior del ISP, de modo que los enrutadores del extremo opuesto del ISP sepan qué enrutador usar para salir del ISP por el otro lado. Podemos ver esto en las rutas de más a la izquierda, en donde el siguiente salto apunta a un enrutador en el mismo ISP y no a un enrutador en el siguiente ISP.

Ahora podemos describir la pieza clave faltante, que es la forma en que los enrutadores BGP eligen la ruta a usar para cada destino. Cada enrutador BGP puede conocer una ruta para un destino dado por medio del enrutador al que está conectado en el siguiente ISP y de todos los demás enrutadores de límite (que han escuchado distintas rutas de los enrutadores conectados a otros ISP). Cada enrutador debe decidir cuál ruta de este conjunto de rutas es la mejor que puede usar. En última instancia, la respuesta es que el ISP debe escribir una política para elegir la ruta preferida. Sin embargo, esta explicación es muy general y no del todo satisfactoria, por lo que al menos podemos describir algunas estrategias comunes.

La primera estrategia es seleccionar las rutas a través de redes entre iguales en vez de las rutas a través de los proveedores de tránsito. Las primeras son gratuitas; las segundas cuestan dinero. Una estrategia similar es dar a las rutas de los clientes la mayor preferencia. Es un buen negocio enviar tráfico directamente a los clientes que pagan.

Un tipo distinto de estrategia es la regla predeterminada que establece que las rutas AS más cortas son mejores. Es algo debatible, puesto que un AS podría ser una red de cualquier tamaño, por lo que una ruta a través de tres pequeños sistemas autónomos podría en realidad ser más corta que una ruta a través de un AS extenso. Sin embargo, lo más corto tiende a ser mejor en promedio, y esta regla es un punto de desempate común.

La estrategia final es preferir la ruta que tenga el menor costo dentro del ISP. Ésta es la estrategia que se implementa en la figura 5-68. Los paquetes enviados de *A* a *C* salen de *AS1* por el enrutador superior, *R1a*. Los paquetes enviados desde *B* salen a través del enrutador inferior, *R1b.* La razón es que tanto *A* como *B* están tomando la ruta de menor costo, o la ruta más rápida, para salir de *AS1*. Como se encuentran en distintas partes del ISP, la salida más rápida para cada uno es distinta. Lo mismo ocurre cuando los paquetes pasan a través de *AS2*. En el último tramo, *AS3* tiene que transportar el paquete proveniente de *B* a través de su propia red.

Esta estrategia se conoce como **enrutamiento de salida anticipada** o **enrutamiento de la papa caliente**. Tiene el curioso efecto colateral de tender a crear rutas asimétricas. Por ejemplo, considere la ruta que se toma cuando *C* envía un paquete de vuelta a *B*. El paquete saldrá rápidamente de *AS3*, en el enrutador superior, para evitar desperdiciar sus recursos. Asimismo, permanecerá en la parte superior cuando *AS2* pase el paquete a *AS1* lo más rápido que pueda. Después el paquete tendrá un viaje más largo en *AS1*. Ésta es una imagen espejo de la ruta que se toma de *B* a *C*.

La discusión anterior debería dejar en claro que cada enrutador BGP selecciona su propia mejor ruta a partir de las posibilidades conocidas. No es el caso, como tal vez se piense ingenuamente, que BGP seleccione una ruta a seguir en el nivel de AS y OSPF seleccione rutas dentro de cada uno de los sistemas autónomos. BGP y el protocolo de puerta de enlace interior están integrados de una manera mucho más profunda. Esto significa que, por ejemplo, BGP puede encontrar el mejor punto de salida de un ISP al siguiente y este punto variará a lo largo del ISP, como es el caso en la política de la papa caliente. También significa que los enrutadores BGP en distintas partes de un AS pueden elegir distintas rutas AS para llegar al mismo destino. El ISP debe tener cuidado al configurar todos los enrutadores BGP para hacer elecciones compatibles teniendo en cuenta toda esta libertad, pero se puede hacer esto en la práctica.

Aunque parezca sorprendente, sólo hemos arañado la superficie de BGP. Para obtener más información, consulte la especificación de la versión 4 de BGP en el RFC 4271 y los RFC relacionados. Sin embargo, cabe mencionar que la mayor parte de su complejidad está en las políticas, las cuales no se describen en la especificación del protocolo BGP.

5.6.8 Multidifusión de Internet

La comunicación normal de IP está entre un emisor y un receptor. Sin embargo, para algunas aplicaciones es útil que un proceso pueda enviar a una gran cantidad de receptores en forma simultánea. Algunos ejemplos son: transmitir por flujo continuo un evento deportivo para muchos espectadores, entregar actualizaciones de programas a una reserva de servidores replicados y manejar llamadas telefónicas de conferencias digitales (es decir, entre varios participantes).

IP apoya la comunicación de uno a varios, o multidifusión, mediante el uso de direcciones IP clase D. Cada dirección clase D identifica a un grupo de hosts. Hay 28 bits disponibles para identificar a los grupos, de modo que pueden existir al mismo tiempo más de 250 millones de grupos. Cuando un proceso envía un paquete a una dirección clase D, se hace el mejor esfuerzo por entregarlo a todos los miembros del grupo direccionado, pero no se da garantía alguna. Quizá algunos miembros no reciban el paquete.

El rango de direcciones IP 224.0.0.0/24 está reservado para multidifusión en la red local. En este caso no se necesita un protocolo de enrutamiento. Para enviar los paquetes por multidifusión, simplemente se

difunden en la LAN con una dirección de multidifusión. Todos los hosts en la LAN reciben las difusiones y los hosts que pertenecen al grupo procesan el paquete. Los enrutadores no reenvían el paquete fuera de la LAN. Algunos ejemplos de direcciones de multidifusión locales son:

224.0.0.1	Todos los sistemas en una LAN.
224.0.0.2	Todos los enrutadores en una LAN.
224.0.0.5	Todos los enrutadores OSPF en una LAN.
224.0.0.251	Todos los servidores DNS en una LAN.

Otras direcciones de multidifusión pueden tener miembros en distintas redes. En este caso se necesita un protocolo de enrutamiento. Pero primero, los enrutadores multidifusión necesitan saber qué hosts son miembros de un grupo. Un proceso pide a su host que se una a un grupo específico. También puede pedir a su host que salga del grupo. Cada host lleva el registro de los grupos a los que pertenecen actualmente sus procesos. Cuando el último proceso de un host sale de un grupo, el host deja de ser miembro de ese grupo. Aproximadamente una vez por minuto, cada enrutador multidifusión envía un paquete de consulta a todos los hosts en su LAN (mediante la dirección de multidifusión local 224.0.0.1, por supuesto) y les pide que se reporten de vuelta a los grupos a los que pertenecen en la actualidad. Los enrutadores multidifusión pueden colocarse o no con los enrutadores estándar. Cada host envía respuestas de vuelta a todas las direcciones de clase D en las que está interesado. Estos paquetes de consulta y respuesta usan un protocolo llamado **IGMP** (**Protocolo de Administración de Grupo de Internet**, del inglés *Internet Group Management Protocol*), que se describe en el RFC 3376.

Se puede usar cualquiera de varios protocolos de enrutamiento multidifusión para construir árboles de expansión de multidifusión que proporcionen rutas de los emisores a todos los miembros del grupo. Los algoritmos que se utilizan son los que describimos en la sección 5.2.8. Dentro de un AS, el protocolo principal que se utiliza es **PIM** (**Multidifusión Independiente del Protocolo**, del inglés *Protocol Independent Multicast*). Hay varios tipos de PIM. En el PIM en modo denso, se crea un árbol de reenvío por ruta inversa recortado. Éste es adecuado para los casos en que los miembros están en todas partes en la red, como al distribuir archivos a muchos servidores dentro de la red de un centro de datos. En el PIM en modo disperso, los árboles de expansión que se construyen son similares a los árboles de núcleo. Esto es adecuado para los casos en que un proveedor de contenido transmite por multidifusión la señal de TV a los suscriptores en su red IP. Una variante de este diseño, conocida como PIM de multidifusión específico del origen, está optimizada para el caso en que sólo hay un emisor para el grupo. Por último, hay que usar extensiones multidifusión para BGP o túneles para poder crear rutas multidifusión cuando los miembros en el grupo se encuentran en más de un AS.

5.6.9 IP móvil

Muchos usuarios de Internet tienen computadoras móviles y desean permanecer conectados cuando están lejos de su hogar, e incluso durante el camino. Por desgracia, con el sistema de direccionamiento de IP, trabajar lejos de casa es más fácil de decir que de hacer, como veremos en breve. De todas formas, cuando las personas empezaron a exigir esta capacidad, la IETF estableció un grupo de trabajo para encontrar una solución. El grupo de trabajo formuló con rapidez varias metas consideradas como deseables en cualquier solución. Las principales fueron:

1. Cada host móvil debía ser capaz de usar su dirección IP base en cualquier parte.
2. No se permitían cambios de software en los hosts fijos.
3. No se permitían cambios en el software ni en las tablas del enrutador.

4. La mayoría de paquetes para host móviles no debían desviarse de la ruta.
5. No se debía incurrir en sobrecarga cuando un host móvil estuviera en casa.

La solución que se eligió se describe en la sección 5.2.10. En resumen, cada sitio que desee permitir a sus usuarios vagar tiene que crear un ayudante en el sitio, conocido como **agente de base**. Cuando aparece un host móvil en un sitio foráneo, obtiene una nueva dirección IP (conocida como dirección de custodia) del sitio foráneo. Después el móvil indica al agente de base su ubicación actual, para lo cual le proporciona la dirección de custodia. Cuando llega un paquete para el móvil en el sitio base y éste se encuentra en otra parte, el agente de base agarra el paquete y lo envía a través de un túnel al móvil, a la dirección actual de custodia. El móvil puede enviar paquetes de respuesta directamente a cualquiera con quien se esté comunicando, pero sigue usando su dirección base como la dirección de origen. Esta solución cumple con todos los requerimientos indicados antes, excepto que los paquetes para los hosts móviles sí se desvían.

Ya que cubrimos la capa de red de Internet, podemos analizar la solución con más detalle. La necesidad de soporte móvil en primer lugar proviene del mismo esquema de direccionamiento IP. Cada dirección IP contiene un número de red y un número de host. Por ejemplo, considere la máquina con la dirección IP 160.80.40.20/16. La parte 160.80 representa el número de red; la parte 40.20 es el número de host. Los enrutadores de todo el mundo tienen tablas de enrutamiento, las cuales les indican qué enlace deben usar para llegar a la red 160.80. Cada vez que llega un paquete con una dirección IP de destino de la forma 160.80.xxx.yyy, pasa por esa línea. Si de repente la máquina con esa dirección se transporta a un sitio distante, los paquetes para ella se seguirán enrutando hacia su LAN (o enrutador) base.

En esta etapa hay dos opciones, aunque ambas son poco atractivas. La primera es que podríamos crear una ruta hacia un prefijo más específico. Es decir, si el sitio distante anuncia una ruta a 160.80.40.20/32, los paquetes que se envíen al destino empezarán a llegar al lugar correcto otra vez. Esta opción depende del algoritmo del prefijo más largo coincidente que se utiliza en los enrutadores. Sin embargo, hemos agregado una ruta a un prefijo IP con una sola dirección IP en él. Todos los ISP en el mundo conocerán este prefijo. Si todos cambiaran las rutas IP globales de esta forma al mover su computadora, cada enrutador tendría millones de entradas en la tabla, con un costo astronómico para Internet. Esta opción no es funcional.

La segunda opción es cambiar la dirección IP del móvil. Es cierto, los paquetes que se envíen a la dirección IP base ya no se entregarán sino hasta que se informa a todas las personas, programas y bases de datos relevantes sobre el cambio. Pero el móvil todavía puede usar Internet en la nueva ubicación para navegar en la Web y ejecutar otras aplicaciones. Esta opción maneja la movilidad en una capa superior. Es lo que ocurre comúnmente cuando un usuario lleva su computadora portátil a una cafetería y usa Internet a través de la red inalámbrica local. La desventaja es que interrumpe algunas aplicaciones y no mantiene la conectividad mientras el equipo se desplaza de un lado a otro.

Como observación adicional, la movilidad también se puede manejar en una capa inferior: la capa de enlace. Esto es lo que ocurre al usar una computadora portátil en una sola red inalámbrica 802.11. La dirección IP del móvil no cambia y la ruta de red permanece igual. Es el enlace inalámbrico quien proporciona la movilidad. Sin embargo, el grado de movilidad es limitado. Si la computadora portátil se aleja demasiado, tendrá que conectarse a Internet por medio de otra red, con una dirección IP distinta.

La solución de IP móvil para IPv4 se proporciona en el RFC 3344. Funciona con el enrutamiento de Internet existente y permite a los hosts permanecer conectados con sus propias direcciones IP a medida que se desplazan. Para que funcione, el móvil debe ser capaz de detectar cada vez que se mueve. Para ello se utiliza el anuncio del enrutador ICMP y los mensajes de solicitud. Los móviles escuchan los anuncios periódicos del enrutador o envían una solicitud para descubrir el enrutador más cercano. Si este enrutador

no tiene la dirección usual del enrutador cuando el móvil está en su base, debe estar en una red foránea. Si este enrutador cambió desde la última vez, el móvil se desplazó a otra red foránea. Este mismo mecanismo permite a los hosts móviles encontrar a sus agentes de base.

Para obtener una dirección IP de custodia en la red foránea, un móvil puede simplemente usar DHCP. Como alternativa, si hay una escasez de direcciones IPv4, el móvil puede enviar y recibir paquetes a través de un agente foráneo que ya tenga una dirección IP en la red. Para buscar un agente foráneo, el host móvil usa el mismo mecanismo ICMP que se utiliza para buscar el agente de base. Una vez que el móvil obtiene una dirección IP o encuentra un agente foráneo, puede usar la red para enviar un mensaje a su agente de base e informarle sobre su ubicación actual.

El agente de base necesita una manera de interceptar los paquetes que se envían al móvil, sólo cuando éste no está en su base. ARP ofrece un mecanismo conveniente para ello. Para enviar un paquete a través de una red Ethernet a un host IP, el enrutador necesita conocer la dirección Ethernet del host. El mecanismo usual es que el enrutador envíe una consulta ARP para preguntar, por ejemplo, cuál es la dirección Ethernet de 160.80.40.20. Cuando el móvil está en su base, responde a las consultas ARP destinadas a su dirección IP con su propia dirección Ethernet. Cuando el móvil está fuera de su base, el agente de base responde a esta consulta, para lo cual proporciona su dirección Ethernet. A continuación el enrutador envía los paquetes para 160.80.40.20 al agente de base. Recuerde que a esto se le denomina ARP de proxy.

Para actualizar de forma rápida las asociaciones ARP de un lado a otro, cuando el móvil sale de su base o llega a ella, se puede usar otra técnica ARP conocida como **ARP gratuito**. Básicamente, el móvil o el agente de base se envían a sí mismos una consulta ARP para la dirección IP móvil que suministre la respuesta correcta, de modo que el enrutador detecte y actualice su asociación.

La tunelización para enviar un paquete entre el agente de base y el host móvil en la dirección de custodia se lleva a cabo mediante el encapsulamiento del paquete con otro encabezado IP destinado para la dirección de custodia. Cuando el paquete encapsulado llega a la dirección de custodia, se elimina el encabezado IP exterior para revelar el paquete.

Al igual que con muchos protocolos de Internet, el secreto está en los detalles; con más frecuencia, en los detalles de la compatibilidad con otros protocolos que se implementan. Existen dos complicaciones. En primer lugar, las cajas NAT necesitan hurgar más allá del encabezado IP para analizar el encabezado TCP o UDP. La forma original de tunelización para IP móvil no usaba estos encabezados, por lo que no funcionaba con cajas NAT. La solución era cambiar el encapsulamiento para incluir un encabezado UDP.

La segunda complicación es que algunos ISP verifican las direcciones IP de origen de los paquetes para ver que coincidan en donde el protocolo de enrutamiento cree que se debería encontrar el origen. Esta técnica se denomina **filtrado de ingreso** y es una medida de seguridad con el propósito de descartar tráfico con direcciones aparentemente incorrectas que pueden ser maliciosas. Sin embargo, los paquetes que se envían del móvil a otros hosts de Internet cuando se encuentra en una red foránea tendrán una dirección IP de origen que estará fuera de lugar, por lo que se descartarán. Para resolver este problema, el móvil puede usar la dirección de custodia como una fuente para enviar mediante un túnel los paquetes de vuelta al agente de base. A partir de aquí, se pueden enviar a Internet desde lo que parece ser la ubicación correcta. El costo es que la ruta es más indirecta.

La seguridad es otro aspecto que no hemos examinado. Cuando un agente de base recibe un mensaje en el que se le pide que reenvíe todos los paquetes de Roberta a cierta dirección IP, es mejor que no acceda a menos que esté convencido de que Roberta es la fuente de esta petición, y no alguien tratando de hacerse pasar por ella. Para este fin se utilizan los protocolos de autentificación criptográficos, que estudiaremos en el capítulo 8.

Los protocolos de movilidad para IPv6 se basan en los fundamentos de IPv4. El esquema anterior sufre del problema de enrutamiento en triángulo, en donde los paquetes que se envían al móvil toman una ruta errónea a través de un agente de base distante. En IPv6 se utiliza la optimización de rutas para seguir una ruta directa entre el móvil y las otras direcciones IP, una vez que los paquetes iniciales han seguido la ruta larga. El IPv6 móvil se define en el RFC 3775.

Hay otro tipo de movilidad que también se está definiendo para Internet. Algunos aviones cuentan con redes inalámbricas integradas, que los pasajeros pueden usar para conectar sus computadoras portátiles a Internet. El avión tiene un enrutador que se conecta con el resto de Internet a través de un enlace inalámbrico (¿acaso esperaba un enlace cableado?). Por lo tanto, ahora tenemos un enrutador volador, lo cual significa que toda la red es móvil. Los diseños de movilidad de red soportan esta situación sin que las computadoras portátiles se den cuenta de que el avión es móvil. En cuanto a lo que a ellas concierne, es sólo otra red más. Puesto que algunas de las computadoras portátiles pueden usar el IP móvil para mantener sus direcciones base mientras están en el avión, entonces tenemos dos niveles de movilidad. La movilidad de red para IPv6 se define en el RFC 3963.

5.7 RESUMEN

La capa de red proporciona servicios a la capa de transporte. Se puede basar en datagramas o en circuitos virtuales. En ambos casos, su tarea principal es enrutar paquetes de la fuente al destino. En las redes de datagramas, se toma una decisión de enrutamiento en cada paquete; y en las de circuitos virtuales se toma al momento de establecer el circuito.

En las redes de computadoras se utilizan muchos algoritmos de enrutamiento. La inundación es un algoritmo simple para enviar un paquete a lo largo de todas las rutas. La mayoría de los algoritmos buscan la ruta más corta y se adaptan a los cambios en la topología de la red. Los principales algoritmos son el enrutamiento por vector de distancia y el enrutamiento de estado del enlace. La mayoría de las redes actuales usan uno de estos dos algoritmos. Otros tópicos de enrutamiento importantes son el uso de la jerarquía en redes extensas, el enrutamiento para hosts móviles, y los enrutamientos de difusión, multidifusión y anycast.

Las redes se pueden congestionar con facilidad, lo cual aumenta el retardo y la pérdida de paquetes. Para de evitar la congestión, los diseñadores de redes diseñan la red de modo que tenga suficiente capacidad, que seleccione las rutas descongestionadas, se rehuse a aceptar más tráfico que indique a las fuentes que reduzcan su velocidad y que se desprenda de la carga.

El siguiente paso que va más allá de sólo tratar con la congestión es realmente intentar alcanzar una calidad de servicio prometida. Algunas aplicaciones se preocupan más por la velocidad de transmisión real, mientras que otras se preocupan más por el retardo y la variación de éste. Los métodos que se pueden utilizar para proporcionarr distintas calidades de servicio incluyen una combinación del modelado de tráfico, la reservación de recursos en los enrutadores y el control de admisión. Entre los métodos que se han diseñado para una buena calidad del servicio se encuentran los servicios integrados de la IETF (incluyendo RSVP) y los servicios diferenciados.

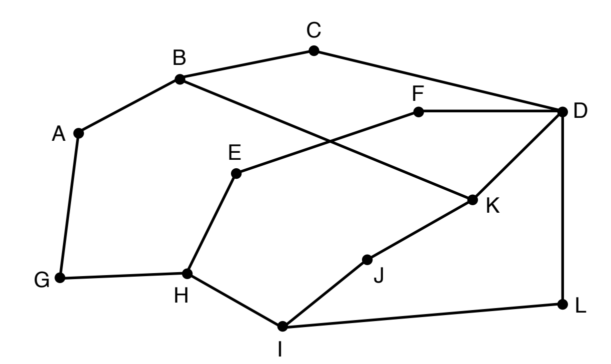
Las redes difieren de varias maneras, por lo que cuando se interconectan múltiples redes pueden ocurrir problemas. Cuando las distintas redes tienen diferentes tamaños máximos de paquete, tal vez se requiera una fragmentación. Las distintas redes pueden ejecutar diferentes protocolos de enrutamiento en forma interna, pero necesitan ejecutar un protocolo común en el exterior. Algunas veces los problemas se pueden refinar mediante la tunelización de un paquete a través de una red hostil, pero si las redes de origen y de destino son diferentes, este método falla.

Internet posee una extensa variedad de protocolos relacionados con la capa de red. Éstos incluyen el protocolo de datagramas IP y los protocolos de control asociados como ICMP, ARP y DHCP. Hay un protocolo orientado a conexión llamado MPLS, el cual transporta paquetes IP a través de algunas redes. Uno de los principales protocolos de enrutamiento que se utilizan dentro de las redes es OSPF; el protocolo de enrutamiento que se utiliza a través de varias redes es BGP. Internet se está quedando rápidamente sin direcciones IP, por lo que se ha desarrollado una versión nueva de IP, conocida como IPv6, pero se está implementando con mucha lentitud.

PROBLEMAS

1. Mencione dos ejemplos de aplicaciones de computadora para las cuales es adecuado un servicio orientado a conexión. Luego mencione dos ejemplos en los que el servicio sin conexiones sea lo mejor.
2. Las redes de datagramas enrutan cada paquete como unidad separada, independiente de las demás. Las redes de circuitos virtuales no tienen que hacer esto, ya que cada paquete de datos sigue una ruta predeterminada. ¿Significa esto que las redes de circuitos virtuales no necesitan la capacidad de enrutar paquetes aislados de una fuente arbitraria a un destino arbitrario? Explique su respuesta.
3. Proporcione tres ejemplos de parámetros de protocolo que podrían negociarse al establecer una conexión.
4. Suponiendo que todos los enrutadores y hosts están trabajando de manera adecuada y que el software de ambos está libre de errores, ¿hay alguna posibilidad, por pequeña que sea, de que un paquete se entregue en el destino equivocado?
5. Proporcione una heurística sencilla para encontrar dos rutas a través de una red, de una fuente dada a un destino dado, que pueda sobrevivir a la pérdida de cualquier línea de comunicación (suponiendo que existen dos de esas rutas). Los enrutadores se consideran lo bastante confiables, por lo que no es necesario preocuparse por la posibilidad de que fallen.
6. Considere la red de la figura 5-12(a). Se usa enrutamiento por vector de distancia y acaban de llegar los siguientes vectores al enrutador *C*: de *B*: (5, 0, 8, 12, 6, 2); de *D*: (16, 12, 6, 0, 9, 10), y de *E*: (7, 6, 3, 9, 0, 4). El costo de los enlaces de *C* a *B*, *D* y *E* son 6, 3 y 5, respectivamente. ¿Cuál es la nueva tabla de enrutamiento de C? Indique tanto la línea de salida a usar como el costo.
7. Si en una red de 50 enrutadores los costos se registran como números de 8 bits y se intercambian vectores de distancia dos veces por segundo, ¿cuánto ancho de banda por línea (*full-dúplex*) consume el algoritmo de enrutamiento distribuido? Suponga que cada enrutador tiene tres líneas que van a los demás enrutadores.
8. En la figura 5-13 el OR booleano de los dos conjuntos de bits ACF es de 111 en cada fila. ¿Es éste un mero accidente, o es cierto para todas las redes en todas las circunstancias?
9. Para un enrutamiento jerárquico con 4 800 enrutadores, ¿qué región y tamaños de clúster debemos elegir para minimizar el tamaño de la tabla de enrutamiento para una jerarquía de tres capas? Un buen lugar de inicio es la hipótesis de que una solución con *k* clústeres de *k* regiones de *k* enrutadores está cerca de ser óptima, lo cual significa que *k* es aproximadamente la raíz cúbica de 4 800 (cerca de 16). Utilice la prueba y el error para verificar las combinaciones en las que los tres parámetros están en el límite de 16.
10. En el texto se indicó que, cuando un host móvil no está en la base, los paquetes enviados a su LAN base son interceptados por su agente de base en esa LAN. En una red IP de una LAN 802.3, ¿cómo logra esta intercepción el agente de base?
11. Dada la red de la figura 5-6 ¿cuántos paquetes se generan por una difusión de *B*, usando:
    (a) el reenvío por ruta invertida?
    (b) el árbol sumidero?
12. Considere la red de la figura 5-15(a). Imagine que entre *F* y *G* se agrega una línea nueva pero el árbol sumidero de la figura 5-15(b) permanece sin cambios. ¿Qué cambios ocurren en la figura 5-15(c)?

13. Calcule un árbol de expansión multidifusión para el enrutador *C* en la siguiente red, para un grupo con miembros en los enrutadores *A*, *B*, *C*, *D*, *E*, *F*, *I* y *K*.



14. Suponga que el nodo *B* de la figura 5-20 ha reiniciado y no tiene información de enrutamiento en sus tablas. De repente necesita una ruta a *H*. Envía difusiones con *TtL* establecido en 1, 2, 3, etcétera. ¿Cuántas rondas se requieren para encontrar una ruta?

15. Como posible mecanismo de control de congestión en una red que usa circuitos virtuales internamente, un enrutador podría abstenerse de confirmar la recepción de un paquete hasta que: (1) sepa que su última transmisión por el circuito virtual se recibió con éxito, y (2) tiene un búfer libre. Para simplificar, suponga que los enrutadores usan un protocolo de parada y espera, y que cada circuito virtual tiene un búfer dedicado a él para cada destino del tráfico. Si se requieren *T* seg para transmitir un paquete (de datos o de confirmación de recepción) y hay *n* enrutadores en la ruta, ¿cuál es la tasa de transmisión con que se entregan paquetes al host de destino? Suponga que los errores de transmisión son poco frecuentes y que la conexión host-enrutador es infinitamente rápida.

16. Una red de datagramas permite que los enrutadores descarten paquetes cuando lo necesiten. La probabilidad de que un enrutador descarte un paquete es *p*. Considere el caso de un host fuente conectado al enrutador fuente, que está conectado al enrutador de destino, y por él al host de destino. Si cualquiera de los enrutadores descarta un paquete, el tiempo del host fuente expirará tarde o temprano e intentará de nuevo. Si tanto las líneas host-enrutador como enrutador-enrutador se cuentan como saltos, ¿cuál es el número promedio de:
(a) saltos que da un paquete por transmisión?
(b) transmisiones que hace un paquete?
(c) saltos requeridos por paquete recibido?

17. Describa dos diferencias principales entre el método ECN y el método RED para evitar congestión.

18. Un esquema de cubeta con tokens se utiliza para modelar tráfico. Se pone un token nuevo en la cubeta cada 5 μseg. Cada token es válido para un paquete corto, que contiene 48 bytes de datos ¿Cuál es la tasa de datos máxima sustentable?

19. Una computadora de una red de 6 Mbps se regula mediante una cubeta con tokens. La cubeta con tokens se llena a una tasa de 1 Mbps. En un principio está llena a su capacidad máxima de 8 megabits. ¿Durante cuánto tiempo puede transmitir la computadora a la máxima velocidad de 6 Mbps?

20. La red de la figura 5-34 utiliza RSVP con árboles de multidifusión para los hosts 1 y 2. Suponga que el host 3 solicita un canal con un ancho de banda de 2 MB/seg para un flujo del host 1 y otro canal con un ancho de banda de 1 MB/seg para un flujo del host 2. Al mismo tiempo, el host 4 solicita un canal con un ancho de banda de 2 MB/seg para un flujo del host 1 y el host 5 solicita un canal con un ancho de banda de 1 MB/seg para un flujo del host 2. ¿Cuánto ancho de banda se reservará en total para estas solicitudes en los enrutadores *A*, *B*, *C*, *E*, *H*, *J*, *K* y *L*?

21. Un enrutador puede procesar 2 millones de paquetes/seg. La carga que se le ofrece es 1.5 millones de paquetes/seg en promedio. Si una ruta del origen al destino contiene 10 enrutadores, ¿cuánto tiempo invierte el enrutador en el encolamiento y en dar servicio?

22. Considere al usuario de los servicios diferenciados con reenvío expedito. ¿Hay alguna garantía de que los paquetes expeditos experimenten un retardo más corto que los paquetes regulares? ¿Por qué sí o por qué no?

23. Suponga que el host *A* está conectado a un enrutador *R1*, que *R1* está conectado a otro enrutador, *R2*, y que *R2* está conectado al host *B*. Suponga que un mensaje TCP que contiene 900 bytes de datos y 20 bytes de encabezado TCP se pasa al código IP en el host *A* para entregarlo a *B*. Muestre los campos *Longitud total*, *Identificación*, *DF*, *MF* y *Desplazamiento del fragmento* del encabezado IP en cada paquete transmitido a través de los tres enlaces. Suponga que el enlace *A-R1* puede soportar un tamaño máximo de trama de 1024 bytes, incluyendo un encabezado de trama de 14 bytes; el enlace *R1-R2* puede soportar un tamaño máximo de trama de 512 bytes, incluyendo un encabezado de trama de 8 bytes, y el enlace *R2-B* puede soportar un tamaño máximo de trama de 512 bytes, incluyendo un encabezado de trama de 12 bytes.

24. Un enrutador está transmitiendo a toda velocidad paquetes IP cuya longitud total (datos más encabezado) es de 1024 bytes. Suponiendo que los paquetes vivan por 10 seg, ¿cuál es la velocidad máxima de línea a la que el enrutador puede operar sin peligro de desbordar el espacio de números de ID de datagramas IP?

25. Un datagrama IP que utiliza la opción *Enrutamiento estricto desde el origen* se tiene que fragmentar. ¿Cree usted que la opción se copia en cada fragmento, o que basta con colocarlo en el primer fragmento? Explique su respuesta.

26. Suponga que en lugar de usar 16 bits para la parte de red de una dirección clase B, se hubieran usado 20 bits. ¿Cuántas redes clase B habría?

27. Convierta la dirección IP cuya representación hexadecimal es C22F1582 a notación decimal con puntos.

28. Una red en Internet tiene una máscara de subred de 255.255.240.0. ¿Cuál es la cantidad máxima de hosts que puede manejar?

29. Mientras que las direcciones IP están atadas a redes específicas, las direcciones Ethernet no. ¿Puede pensar en una buena razón por la que no lo están?

30. Hay una gran cantidad de direcciones IP consecutivas diponibles comenzando con 198.16.0.0. Suponga que cuatro organizaciones, *A*, *B*, *C* y *D*, solicitan 4 000, 2 000, 4 000 y 8 000 direcciones, respectivamente y en ese orden. Para cada una de ellas, proporcione la primera dirección IP asignada, la última dirección IP asignada y la máscara en la notación *w.x.y.z/s*.

31. Un enrutador acaba de recibir las siguientes direcciones IP NUEVAS: 57.6.96.0/21, 57.6.104.0/21, 57.6.112.0/21 y 57.6.120.0/21. Si todas éstas utilizan la misma línea de salida, ¿se pueden agregar? De ser así, ¿a qué? Si no, ¿por qué?

32. El conjunto de direcciones IP de 29.18.0.0 a 19.18.128.255 se ha agregado a 29.18.0.0/17. Sin embargo, hay un hueco de 1024 direcciones sin asignar de 29.18.60.0 a 29.18.63.255 que de repente se asignan a un host que utiliza una línea de salida diferente. ¿Es ahora necesario dividir la dirección agregada en sus bloques constituyentes, agregar el nuevo bloque a la tabla y, después, ver si es posible alguna reagregación? Si no lo es, ¿qué se puede hacer en lugar de eso?

33. Un enrutador tiene las siguientes entradas (CIDR) en su tabla de enrutamiento:

Dirección/máscara	**Siguiente salto**
135.46.56.0/22	Interfaz 0
135.46.60.0/22	Interfaz 1
192.53.40.0/23	Enrutador 1
Predeterminada	Enrutador 2

Para cada una de las siguientes direcciones IP, ¿qué hace el enrutador si llega un paquete con esa dirección?

(a) 135.46.63.10
(b) 135.46.57.14
(c) 135.46.52.2
(d) 192.53.40.7
(e) 192.53.56.7

34. Muchas compañías tienen la política de contar con dos (o más) enrutadores que conecten la compañía a Internet para proporcionar cierta redundancia en caso de que uno de ellos falle. ¿Esta política aún es posible con NAT? Explique su respuesta.

35. Usted acaba de explicar el protocolo ARP a un amigo. Al terminar su explicación, él le dice: “Ya entiendo. ARP proporciona un servicio a la capa de red, por lo que es parte de la capa de enlace de datos”. ¿Qué le diría a su amigo?

36. Describa una forma de reensamblar fragmentos IP en el destino.

37. La mayoría de los algoritmos de reensamble de datagramas IP tienen un temporizador para evitar que un fragmento perdido ocupe búferes de reensamble por siempre. Suponga que un datagrama se divide en cuatro fragmentos. Los primeros tres fragmentos llegan pero el cuarto se retrasa. En algún momento, el temporizador expira y se descartan los tres fragmentos de la memoria del receptor. Un poco más tarde, llega el último fragmento. ¿Qué se debería hacer con él?

38. En IP, la suma de verificación cubre sólo el encabezado y no los datos. ¿Por qué supone que se eligió este diseño?

39. Una persona que vive en Boston viaja a Minneapolis, y lleva consigo su computadora portátil. Para su sorpresa, la LAN de su destino en Minneapolis es una LAN IP inalámbrica, por lo que no tiene que conectarse. ¿Es aún necesario pasar por todo el proceso de los agentes de base y agentes foráneos para que el correo electrónico y otro tipo de tráfico llegue en forma correcta?

40. IPv6 utiliza direcciones de 16 bytes. Si se asigna un bloque de 1 millón de direcciones cada picosegundo, ¿cuánto durarán las direcciones?

41. El campo *Protocolo* utilizado en el encabezado IPv4 no está presente en el encabezado IPv6 fijo. ¿Por qué no?

42. Cuando se introduzca el protocolo IPv6, ¿habrá que cambiar el protocolo ARP? De ser así, ¿los cambios serán conceptuales o técnicos?

43. Escriba un programa para simular enrutamiento que utilice inundación. Cada paquete debe contener un contador que se decremente en cada salto. Cuando el contador llegue a cero, hay que descartar el paquete. El tiempo es discreto y cada línea maneja un paquete por intervalo. Cree tres versiones del programa: todas las líneas están inundadas, todas las líneas, excepto la de entrada, están inundadas, y sólo las k mejores líneas (elegidas de manera estática) están inundadas. Compare la inundación con el enrutamiento determinista ($k = 1$) en base al retardo y el ancho de banda utilizado.

44. Escriba un programa que simule una red de computadoras mediante el uso de tiempo discreto. El primer paquete en cada cola de enrutador da un salto por intervalo. Cada enrutador sólo tiene un número finito de búferes. Si un paquete llega y no hay espacio para él, se descarta y no se retransmite. En su lugar hay un protocolo de extremo a extremo, lleno de expiraciones de temporización y paquetes de confirmación de recepción, que en algún momento regeneran dicho paquete del enrutador de origen. Grafique la velocidad de transmisión real de la red como una función del intervalo de expiración de temporizador de extremo a extremo, con parámetros de tasa de error.

45. Escriba una función para realizar el reenvío en un enrutador IP. El procedimiento tiene un parámetro: una dirección IP. También tiene acceso a una tabla global que consiste de un arreglo de tres variables. Cada arreglo contiene tres enteros: una dirección IP, una máscara de subred y la línea de salida a utilizar. La función usa CIDR para buscar la dirección IP en la tabla y devuelve la línea a utilizar como su valor.

46. Utilice los programas *traceroute* (UNIX) o *tracert* (Windows) para trazar la ruta de su computadora a varias universidades de otros continentes. Haga una lista de los enlaces transoceánicos que descubra. Algunos sitios para probar son:

- www.berkeley.edu (California)
- www.mit.edu (Massachusetts)
- www.vu.nl (Ámsterdam)
- www.ucl.ac.uk (Londres)
- www.usyd.edu.au (Sidney)
- www.u-tokyo.ac.jp (Tokio)
- www.uct.ac.za (Cape Town)

6

LA CAPA DE TRANSPORTE

Junto con la capa de red, la capa de transporte es el corazón de la jerarquía de protocolos. La capa de red provee una entrega de paquetes punto a punto mediante el uso de datagramas o circuitos virtuales. La capa de transporte se basa en la capa de red para proveer transporte de datos de un proceso en una máquina de origen a un proceso en una máquina de destino, con un nivel deseado de confiabilidad que es independiente de las redes físicas que se utilizan en la actualidad. Ofrece las abstracciones que necesitan las aplicaciones para usar la red. Sin esta capa, todo el concepto de protocolos por capas tendría muy poco sentido. En este capítulo estudiaremos la capa de transporte con detalle, incluyendo sus servicios y la elección de diseño de la API para lidiar con las cuestiones de confiabilidad, conexiones y control de la congestión, los protocolos como TCP y UDP, y el desempeño.

6.1 EL SERVICIO DE TRANSPORTE

En las siguientes secciones veremos una introducción al servicio de transporte. Analizaremos el tipo de servicio que se proporciona a la capa de aplicación. Para que el tema del servicio de transporte sea más concreto, examinaremos dos conjuntos de primitivas de la capa de transporte. Primero analizaremos uno muy sencillo (pero hipotético) para mostrar las ideas básicas. Después veremos la interfaz que se utiliza comúnmente en Internet.

6.1.1 Servicios que se proporcionan a las capas superiores

La meta fundamental de la capa de transporte es proporcionar un servicio de transmisión de datos eficiente, confiable y económico a sus usuarios, procesos que normalmente son de la capa de aplicación. Para lograr este objetivo, la capa de transporte utiliza los servicios proporcionados por la capa de red. El hardware o software de la capa de transporte que se encarga del trabajo

se llama **entidad de transporte**, la cual puede localizarse en el kernel (núcleo) del sistema operativo, en un paquete de biblioteca que forma parte de las aplicaciones de red, en un proceso de usuario separado o incluso en la tarjeta de interfaz de red. Las primeras dos opciones son las más comunes en Internet. En la figura 6-1 se ilustra la relación (lógica) entre las capas de red, transporte y aplicación.

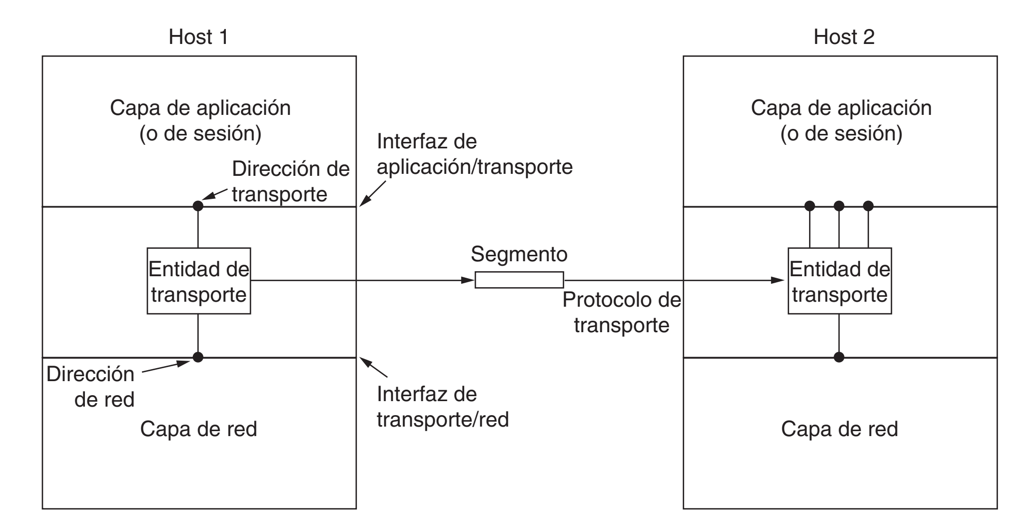


Figura 6-1. Las capas de red, transporte y aplicación.

Así como hay dos tipos de servicio de red, orientado a conexión y sin conexión, también hay dos tipos de servicio de transporte. El que está orientado a conexión es parecido en muchos sentidos al servicio de red orientado a conexión. En ambos casos, las conexiones tienen tres fases: establecimiento, transferencia de datos y liberación. El direccionamiento y el control de flujo también son similares en ambas capas. Además, el servicio de transporte sin conexión es muy parecido al servicio de red sin conexión. Sin embargo, puede ser difícil proveer un servicio de transporte sin conexión encima de un servicio de red orientado a conexión, ya que es ineficiente establecer una conexión para enviar un solo paquete y deshacerla justo después.

La pregunta obvia es: si el servicio de la capa de transporte es tan parecido al de la capa de red, ¿por qué hay dos capas diferentes? ¿Por qué no es suficiente una sola capa? La respuesta es sutil, pero crucial. El código de transporte se ejecuta por completo en las máquinas de los usuarios, pero la capa de red se ejecuta en su mayor parte en los enrutadores, los cuales son operados por la empresa portadora (por lo menos en el caso de una red de área amplia). ¿Qué sucede si la capa de red ofrece un servicio inadecuado? ¿Qué tal si esa capa pierde paquetes con frecuencia? ¿Qué ocurre si los enrutadores fallan de vez en cuando?

Problemas, eso es lo que ocurre. Los usuarios no tienen un control real sobre la capa de red, por lo que no pueden resolver los problemas de un mal servicio usando mejores enrutadores o incrementando el manejo de errores en la capa de enlace de datos, puesto que no son dueños de los enrutadores. La única posibilidad es poner encima de la capa de red otra capa que mejore la calidad del servicio. Si en una red sin conexión se pierden paquetes o se rompen, la entidad de transporte puede detectar el problema y compensarlo mediante el uso de retransmisiones. Si, en una red orientada a conexión, se informa a la entidad de transporte a la mitad de una transmisión extensa que su conexión de red ha sido terminada de manera abrupta, sin indicación de lo que ha sucedido a los datos actualmente en tránsito, la entidad puede establecer una nueva conexión de red con la entidad de transporte remota. A través de esta nueva conexión de red, la entidad puede enviar una solicitud a su igual para preguntarle cuáles datos llegaron y cuáles no, y como sabe en dónde se encontraba, puede reiniciar a partir de donde se originó la interrupción.

En esencia, la existencia de la capa de transporte hace posible que el servicio de transporte sea más confiable que la red subyacente. Además, las primitivas de transporte se pueden implementar como llamadas a procedimientos de biblioteca para que sean independientes de las primitivas de red. Las llamadas al servicio de red pueden variar de manera considerable de una red a otra (por ejemplo, las llamadas basadas en una Ethernet sin conexión pueden ser muy distintas de las llamadas en una red WiMAX orientada a conexión). Al ocultar el servicio de red detrás de un conjunto de primitivas de servicio de transporte, aseguramos que para cambiar la red simplemente hay que reemplazar un conjunto de procedimientos de biblioteca por otro que haga lo mismo con un servicio subyacente distinto.

Gracias a la capa de transporte, los programadores de aplicaciones pueden escribir código de acuerdo con un conjunto estándar de primitivas; estos programas pueden funcionar en una amplia variedad de redes sin necesidad de preocuparse por lidiar con diferentes interfaces de red y distintos niveles de confiabilidad. Si todas las redes reales fueran perfectas, tuvieran las mismas primitivas de servicio y se pudiera garantizar que nunca jamás cambiaran, tal vez la capa de transporte sería innecesaria. Sin embargo, en el mundo real esta capa cumple la función clave de aislar a las capas superiores de la tecnología, el diseño y las imperfecciones de la red.

Por esta razón, muchas personas han establecido una distinción cualitativa entre las capas 1 a 4, por una parte, y la(s) capa(s) por encima de la 4, por la otra. Las cuatro capas inferiores se pueden ver como el **proveedor del servicio de transporte**, mientras que la(s) capa(s) superior(es) son el **usuario del servicio de transporte**. Esta distinción entre proveedor y usuario tiene un impacto considerable en el diseño de las capas, además de que posiciona a la capa de transporte en un puesto clave, ya que constituye el límite principal entre el proveedor y el usuario del servicio confiable de transmisión de datos. Es el nivel que ven las aplicaciones.

6.1.2 Primitivas del servicio de transporte

Para permitir que los usuarios accedan al servicio de transporte, la capa de transporte debe proporcionar algunas operaciones a los programas de aplicación; es decir, una interfaz del servicio de transporte. Cada servicio de transporte tiene su propia interfaz. En esta sección examinaremos primero un servicio de transporte sencillo (hipotético) y su interfaz para ver los aspectos esenciales. En la siguiente sección veremos un ejemplo real.

El servicio de transporte es similar al servicio de red, pero también hay algunas diferencias importantes. La principal es que el propósito del servicio de red es modelar el servicio ofrecido por las redes reales, con todos sus problemas. Las redes reales pueden perder paquetes, por lo que el servicio de red por lo general no es confiable.

En cambio, el servicio de transporte orientado a conexión sí es confiable. Claro que las redes reales no están libres de errores, pero ése es precisamente el propósito de la capa de transporte: ofrecer un servicio confiable en una red no confiable.

Como ejemplo, considere dos procesos en una sola máquina, conectados mediante una canalización en UNIX (o cualquier otra herramienta de comunicación entre procesos). Ambos consideran que la conexión entre ellos es 100% perfecta. No quieren saber de confirmaciones de recepción, paquetes perdidos, congestión ni nada por el estilo. Lo que quieren es una conexión 100% confiable. El proceso *A* pone datos en un extremo de la canalización y el proceso *B* los saca por el otro extremo. Ésta es la esencia del servicio de transporte orientado a conexión: ocultar las imperfecciones del servicio de red para que los procesos de usuarios puedan dar por hecho simplemente la existencia de un flujo de bits libre de errores, incluso cuando estén en distintas máquinas.

Como nota al margen, la capa de transporte también puede proporcionar un servicio no confiable (de datagramas). Sin embargo, hay muy poco que decir al respecto, además del hecho de que “son datagra-

mas”, por lo que en este capítulo nos concentraremos principalmente en el servicio de transporte orientado a conexión. No obstante, hay algunas aplicaciones que se benefician del transporte sin conexión, como la computación cliente-servidor y la multimedia de flujo continuo, por lo que veremos algo sobre ellas más adelante.

Una segunda diferencia entre los servicios de red y de transporte es a quién están dirigidos. El servicio de red lo usan únicamente las entidades de transporte. Pocos usuarios escriben sus propias entidades de transporte y, por lo tanto, pocos usuarios o programas llegan a ver los aspectos internos del servicio de red. En contraste, muchos programas (y, por lo tanto, programadores) ven las primitivas de transporte. En consecuencia, el servicio de transporte debe ser conveniente y fácil de usar.

Para tener una idea de lo que podría ser un servicio de transporte, considere las cinco primitivas listadas en la figura 6-2. Esta interfaz de transporte ciertamente es sencilla, pero muestra la esencia de lo que debe hacer una interfaz de transporte orientada a conexión: permitir que los programas de aplicación establezcan, usen y después liberen las conexiones, lo cual es suficiente para muchas aplicaciones.

Primitiva	Paquete enviado	Significado
LISTEN	(ninguno)	Se bloquea hasta que algún proceso intenta conectarse.
CONNECT	CONNECTION REQ.	Intenta activamente establecer una conexión.
SEND	DATA	Envía información.
RECEIVE	(ninguno)	Se bloquea hasta que llegue un paquete DATA.
DISCONNECT	DISCONNECTION REQ.	Solicita que se libere la conexión

Figura 6-2. Las primitivas para un servicio simple de transporte.

Para ver cómo podrían usarse estas primitivas, considere una aplicación con un servidor y cierta cantidad de clientes remotos. Para empezar, el servidor ejecuta una primitiva LISTEN, por lo general mediante la llamada a un procedimiento de biblioteca que hace una llamada al sistema para bloquear al servidor hasta que aparezca un cliente. Cuando un cliente desea comunicarse con el servidor, ejecuta una primitiva CONNECT. La entidad de transporte ejecuta esta primitiva, para lo cual bloquea al invocador y envía un paquete al servidor. En la carga útil de este paquete se encuentra un mensaje de capa de transporte encapsulado, dirigido a la entidad de transporte del servidor.

Aquí es pertinente una nota rápida sobre la terminología. A falta de un mejor término, usaremos **segmento** para indicar los mensajes que se envían de una entidad de transporte a otra. TCP, UDP y otros protocolos de Internet usan este término. Algunos protocolos antiguos usaban el nombre de **TPDU** (**Unidad de Datos del Protocolo de Transporte**, del inglés *Transport Protocol Data Unit*). Este término ya no se utiliza mucho, pero todavía se puede ver en publicaciones y libros antiguos.

Así, los segmentos (intercambiados por la capa de transporte) están contenidos en paquetes (intercambiados por la capa de red). A su vez, estos paquetes están contenidos en tramas (intercambiadas por la capa de enlace de datos). Cuando llega una trama, la capa de enlace de datos procesa el encabezado de la trama y, si la dirección de destino coincide para la entrega local, pasa el contenido del campo de carga útil de la trama a la entidad de red. Esta última procesa de manera similar el encabezado del paquete y después pasa el contenido de la carga útil del paquete a la entidad de transporte. Este anidamiento se ilustra en la figura 6-3.

Figura 6-3. Anidamiento de segmentos, paquetes y tramas.

Ahora regresemos a nuestro ejemplo de cliente-servidor. La llamada CONNECT del cliente ocasiona el envío de un segmento CONNECTION REQUEST (solicitud de conexión) al servidor. Al llegar este segmento, la entidad de transporte verifica que el servidor esté bloqueado en LISTEN (es decir, que esté interesado en manejar solicitudes). Si es así, entonces desbloquea el servidor y envía un segmento CONNECTION ACCEPTED (conexión aceptada) de regreso al cliente. Al llegar este segmento, el cliente se desbloquea y se establece la conexión.

Ahora pueden intercambiarse datos mediante las primitivas SEND y RECEIVE. En la forma más simple, cualquiera de las dos partes puede emitir un RECEIVE (bloqueo) para esperar que la otra parte emita un SEND. Al llegar el segmento, el receptor se desbloquea. Entonces puede procesar el segmento y enviar una respuesta. Mientras ambos lados puedan llevar el control de quién tiene el turno para transmitir, este esquema funciona bien.

Observe que en la capa de transporte, incluso un intercambio de datos unidireccional es más complicado que en la capa de red. También se confirmará (tarde o temprano) la recepción de cada paquete de datos enviado. Asimismo, la recepción de los paquetes que llevan segmentos de control se confirmará de manera implícita o explícita. Estas confirmaciones se manejan mediante las entidades de transporte usando el protocolo de capa de red, por lo que no son visibles para los usuarios de transporte. De la misma forma, las entidades de transporte tienen que preocuparse por los temporizadores y las retransmisiones. Los usuarios de transporte no se enteran de ningún aspecto de esta mecánica. Para ellos una conexión es un conducto de bits confiable: un usuario mete bits en él y por arte de magia aparecen en el otro lado, con el mismo orden. Esta habilidad de ocultar la complejidad es la razón por la cual los protocolos en capas son herramientas tan poderosas.

Cuando ya no es necesaria una conexión, hay que liberarla para desocupar espacio en las tablas de las dos entidades de transporte. La desconexión tiene dos variantes: asimétrica y simétrica. En la variante asimétrica, cualquiera de los dos usuarios de transporte puede emitir una primitiva DISCONNECT, con lo cual se envía un segmento DISCONNECT a la entidad de transporte remota. A su llegada se libera la conexión.

En la variante simétrica, cada dirección se cierra por separado, independientemente de la otra. Cuando una de las partes emite una primitiva DISCONNECT, quiere decir que ya no tiene más datos por enviar pero aún está dispuesta a recibir datos de la otra parte. En este modelo, una conexión se libera cuando ambas partes han emitido una primitiva DISCONNECT.

En la figura 6-4 se presenta un diagrama de estado para establecer y liberar una conexión para estas sencillas primitivas. Cada transición se activa mediante algún evento, ya sea una primitiva ejecutada por el usuario de transporte local o la llegada de un paquete. Por simplicidad, supongamos que la confirmación de recepción de cada segmento se realiza por separado. También supondremos que se usa un modelo de desconexión simétrica, en donde el cliente se desconecta primero. Cabe señalar que este modelo es muy poco sofisticado. Más adelante analizaremos modelos más realistas cuando describamos la forma en que funciona TCP.

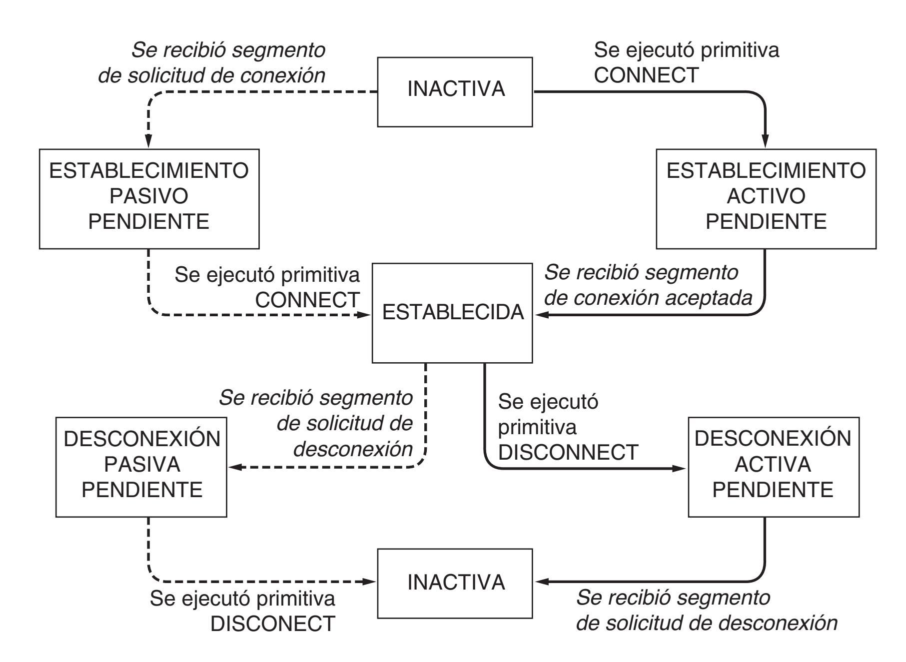


Figura 6-4. Un diagrama de estado para un esquema simple de manejo de conexiones. Las transiciones etiquetadas en cursiva se producen debido a la llegada de paquetes. Las líneas continuas muestran la secuencia de estados del cliente. Las líneas punteadas muestran la secuencia de estados del servidor.

6.1.3 Sockets de Berkeley

Inspeccionemos brevemente otro conjunto de primitivas de transporte: las primitivas de socket que se utilizan para TCP. Los sockets se liberaron primero como parte de la distribución de software de Berkeley UNIX 4.2BSD en 1983. Su popularidad aumentó muy rápido. Las primitivas se utilizan mucho en la actualidad para la programación de Internet en muchos sistemas operativos, en especial los sistemas basados en UNIX; también hay una API estilo sockets para Windows, llamada *winsock*.

Las primitivas se listan en la figura 6-5. En esencia, siguen el modelo de nuestro primer ejemplo pero ofrecen más características y flexibilidad. No analizaremos aquí los segmentos correspondientes. Eso lo haremos después.

Primitiva	Significado
SOCKET	Crea un nuevo punto terminal de comunicación.
BIND	Asocia una dirección local con un socket.
LISTEN	Anuncia la disposición de aceptar conexiones; indica el tamaño de la cola.
ACCEPT	Establece en forma pasiva una conexión entrante.
CONNECT	Intenta establecer activamente una conexión.
SEND	Envía datos a través de la conexión.
RECEIVE	Recibe datos de la conexión.
CLOSE	Libera la conexión.

Figura 6-5. Las primitivas de socket para TCP.

Los servidores ejecutan las primeras cuatro primitivas de la lista en ese orden. La primitiva SOCKET crea un nuevo punto terminal y le asigna espacio en las tablas de la entidad de transporte. Los parámetros de la llamada especifican el formato de direccionamiento que se utilizará, el tipo de servicio deseado (por ejemplo, flujo confiable de bytes) y el protocolo. Una llamada SOCKET con éxito devuelve un descriptor de archivo ordinario que se utiliza con las siguientes llamadas, de la misma manera que lo hace una llamada OPEN.

Los sockets recién creados no tienen direcciones de red. Éstas se asignan mediante la primitiva BIND. Una vez que un servidor ha destinado una dirección a un socket, los clientes remotos se pueden conectar a él. La razón para que la llamada SOCKET no cree directamente una dirección es que algunos procesos se encargan de sus direcciones (por ejemplo, han estado usando su misma dirección durante años y todos la conocen), mientras que otros no lo hacen.

A continuación viene la llamada LISTEN, que asigna espacio para poner en cola las llamadas entrantes por si varios clientes intentan conectarse al mismo tiempo. A diferencia de la llamada LISTEN de nuestro primer ejemplo, en el modelo de sockets LISTEN no es una llamada bloqueadora.

Para bloquearse en espera de una conexión entrante, el servidor ejecuta una primitiva ACCEPT. Cuando llega un segmento que solicita una conexión, la entidad de transporte crea un socket nuevo con las mismas propiedades que el original y devuelve un descriptor de archivo para él. A continuación, el servidor puede ramificar un proceso o hilo para manejar la conexión en el socket nuevo y regresar a esperar la siguiente conexión en el socket original. ACCEPT devuelve un descriptor de archivo que se puede utilizar para leer y escribir de la forma estándar, al igual que con los archivos.

Ahora veamos el lado cliente. Aquí también se debe crear primero un socket mediante la primitiva SOCKET, pero no se requiere BIND puesto que la dirección usada no le importa al servidor. La primitiva CONNECT bloquea al invocador y comienza el proceso de conexión. Al completar este proceso (es decir, cuando se recibe un segmento apropiado del servidor) el proceso cliente se desbloquea y se establece la conexión. Ambos lados pueden usar ahora SEND y RECEIVE para transmitir y recibir datos a través de la conexión full-dúplex. Las llamadas de sistema READ y WRITE de UNIX también se pueden utilizar si no son necesarias las opciones especiales de SEND y RECEIVE.

La liberación de las conexiones a los sockets es simétrica. La conexión se libera cuando ambos lados ejecutan una primitiva CLOSE.

Los sockets se han vuelto muy populares, además de ser el estándar de facto para abstraer servicios de transporte para las aplicaciones. La API de sockets se utiliza comúnmente con el protocolo TCP para ofrecer un servicio orientado a conexión, conocido como **flujo confiable de bytes**, que consiste en el conducto de bits confiable que describimos antes. Sin embargo, se podrían usar otros protocolos para implementar este servicio mediante el uso de la misma API. Debería ser igual para todos los usuarios del servicio de transporte.

Un punto fuerte de la API de sockets es que cualquier aplicación la puede utilizar para otros servicios de transporte. Por ejemplo, se pueden usar sockets con un servicio de transporte sin conexión. En este caso, CONNECT establece la dirección del transporte del igual remoto, mientras que SEND y RECEIVE envían y reciben datagramas hacia/desde el igual remoto (también es común usar un conjunto expandido de llamadas —por ejemplo, SENDTO y RECEIVEFROM— que enfaticen los mensajes y no limiten una aplicación a un solo igual de transporte). Los sockets también se pueden usar con protocolos de transporte que proporcionen un flujo continuo de mensajes en vez de un flujo continuo de bytes, y que dispongan o no de un control de congestión. Por ejemplo, **DCCP** (**Protocolo de Control de Congestión de Datagramas**, del inglés *Datagram Congestion Controlled Protocol*) es una versión de UDP con control de congestión (Kohler y colaboradores, 2006). Depende de los usuarios de transporte comprender qué servicio están recibiendo.

Sin embargo, no es probable que los sockets vayan a ser la última palabra en las interfaces de transporte. Por ejemplo, las aplicaciones trabajan a menudo con un grupo de flujos relacionados, como un

navegador web que solicita varios objetos al mismo servidor. Con los sockets, lo más natural es que los programas de aplicación usen un flujo por cada objeto. Esta estructura significa que el control de congestión se aplica por separado para cada flujo, no entre todo el grupo, lo cual es subóptimo. Esto libra a la aplicación de la carga de tener que administrar el conjunto. Se han desarrollado protocolos e interfaces más recientes que soportan grupos de flujos relacionados de una forma más efectiva y simple para la aplicación. Dos ejemplos son **SCTP** (**Protocolo de Control de Transmisión de Flujo**, del inglés *Stream Control Transmission Protocol*), que se define en el RFC 4960, y **SST** (**Transporte Estructurado de Flujo**, del inglés *Structured Stream Transport*) (Ford, 2007). Estos protocolos deben modificar un poco la API para obtener los beneficios de los grupos de flujos relacionados; además soportan características tales como una mezcla de tráfico orientado a conexión y sin conexión, e incluso múltiples rutas de red. El tiempo dirá si tienen éxito o fracasan.

6.1.4 Un ejemplo de programación de sockets: un servidor de archivos de Internet

Veamos ahora el código de cliente y servidor de la figura 6-6 como ejemplo del uso real de las llamadas de sockets. Ahí se muestra un servidor de archivos de Internet muy primitivo, junto con un cliente de ejemplo que lo utiliza. El código tiene muchas limitaciones (que analizaremos más adelante), pero en principio el código del servidor se puede compilar y ejecutar en cualquier máquina UNIX conectada a Internet y el código del cliente también se puede compilar y ejecutar en cualquier otra máquina UNIX en Internet, en cualquier parte del mundo. El código del cliente se puede ejecutar con los parámetros apropiados para obtener cualquier archivo al que el servidor tenga acceso en su máquina. El archivo se escribe a la salida estándar que, por supuesto, se puede redirigir a un archivo o conducto.

Veamos primero el código del servidor. Comienza con algunos encabezados estándar, los últimos tres contienen las principales definiciones y estructuras de datos relacionadas con Internet. A continuación se encuentra una definición de *SERVER_PORT* como 12345. Este número se eligió de manera arbitraria. Cualquier número que se encuentre entre 1024 y 65535 funcionará siempre y cuando otro proceso no lo esté utilizando; los puertos debajo de 1023 están reservados para los usuarios privilegiados.

Las siguientes líneas del servidor definen dos constantes necesarias. La primera determina el tamaño de bloque en bytes que se utiliza para la transferencia de archivos. La segunda determina cuántas conexiones pendientes se pueden almacenar antes de empezar a descartar las conexiones adicionales que vayan llegando.

```
/* Esta página contiene un programa cliente que puede solicitar un archivo desde el programa servidor de la siguiente
   página. El servidor responde
* enviando el archivo completo.
*/
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define SERVER_PORT 12345                 /* arbitrario, pero el cliente y el servidor deben coincidir */
#define BUF_SIZE 4096                     /* tamaño de bloque para transferencia */

int main(int argc, char **argv)
{
 int c, s, bytes;
 char buf[BUF_SIZE];                      /* búfer para el archivo entrante */
 struct hostent *h;                       /* información sobre el servidor */
 struct sockaddr_in channel;              /* contiene la dirección IP */
```

(*continúa*)

(*continuación*)

```
 if (argc != 3) fatal("Usar: cliente nombre-servidor nombre-archivo");
 h = gethostbyname(argv[1]);                          /* busca la dirección IP del host */
 if (!h) fatal("falla en gethostbyname ");

 s = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
 if (s < 0) fatal("socket");
 memset(&channel, 0, sizeof(channel));
 channel.sin_family= AF_INET;
 memcpy(&channel.sin_addr.s_addr, h->h_addr, h->h_length);
 channel.sin_port= htons(SERVER_PORT);

 c = connect(s, (struct sockaddr *) &channel, sizeof(channel));
 if (c < 0) fatal("falla en conexion");

 /* Se ha establecido la conexión. Se envía el nombre del archivo incluyendo el byte 0 al final. */
 write(s, argv[2], strlen(argv[2])+1);

 /* Obtiene el archivo y lo escribe en la salida estándar. */
 while (1) {
      bytes = read(s, buf, BUF_SIZE);                 /* lee del socket */
      if (bytes <= 0) exit(0);                                 /* verifica el final del archivo */
      write(1, buf, bytes);                                    /* escribe en la salida estándar */
 }
}

fatal(char *string)
{
 printf("%s\n", string);
 exit(1);
}
```

Figura 6-6. Código del cliente que utiliza sockets. El código del servidor está en la siguiente página.

```
#include <sys/types.h>                                /* Éste es el código del servidor */
#include <sys/fcntl.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define SERVER_PORT 12345                             /* arbitrario, pero el cliente y el servidor deben coincidir */
#define BUF_SIZE 4096                                 /* tamaño de bloque para la transferencia */
#define QUEUE_SIZE 10

int main(int argc, char *argv[])
{
 int s, b, l, fd, sa, bytes, on = 1;
 char buf[BUF_SIZE];                                  /* búfer para el archivo saliente */
 struct sockaddr_in channel;                          /* contiene la dirección IP */

 /* Construye la estructura de la dirección para enlazar el socket. */
 memset(&channel, 0, sizeof(channel));                /* canal cero */
 channel.sin_family = AF_INET;
 channel.sin_addr.s_addr = htonl(INADDR_ANY);
 channel.sin_port = htons(SERVER_PORT);

 /* Apertura pasiva. Espera una conexión. */
 s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);       /* crea el socket */
 if (s < 0) fatal("falla en socket ");
 setsockopt(s, SOL_SOCKET, SO_REUSEADDR, (char *) &on, sizeof(on));
```

(*continúa*)

(*continuación*)

```
b = bind(s, (struct sockaddr *) &channel, sizeof(channel));
if (b < 0) fatal("falla en bind ");
l = listen(s, QUEUE_SIZE);                        /* especifica el tamaño de la cola */
if (l < 0) fatal("falla en listen");

/* El socket ahora está configurado y enlazado. Espera una conexión y la procesa. */
while (1) {
    sa = accept(s, 0, 0);                         /* se bloquea para la solicitud de conexión */
    if (sa < 0) fatal("falla en accept");

    read(sa, buf, BUF_SIZE);                      /* lee el nombre del archivo desde el socket */

    /* Obtiene y regresa el archivo.*/
    fd = open(buf, O_RDONLY);                     /* abre el archivo para regresarlo */
    if (fd < 0) fatal("falla en open");

    while (1) {
       bytes = read(fd, buf, BUF_SIZE); /* lee del archivo */
       if (bytes <= 0) break;                     /* verifica el final del archivo */
       write(sa, buf, bytes);                     /* escribe bytes en el socket */
    }
    close(fd);                                    /* cierra el archivo */
    close(sa);                                    /* cierra la conexión */
}
}
```

El código del servidor empieza después de las declaraciones de las variables locales. Primero inicializa una estructura de datos que contendrá la dirección IP del servidor. Esta estructura de datos pronto se anexará al socket del servidor. La llamada a *memset* establece en 0s toda la estructura de datos. Las siguientes tres asignaciones llenarán tres de sus campos. El último de éstos contiene el puerto del servidor. Las funciones *htonl* y *htons* están relacionadas con la conversión de valores a un formato estándar a fin de que el código se ejecute correctamente, tanto en las máquinas *little-endian* (por ejemplo, Intel x86) como en las máquinas big-endian (por ejemplo, la SPARC). Su semántica exacta no es importante aquí.

A continuación el servidor crea un socket y verifica si hay errores (lo cual se indica mediante $s < 0$). En una versión de producción del código, el mensaje de error puede ser un poco más explicativo. La llamada a *setsockopt* es necesaria para permitir la reutilización del puerto, a fin de que el servidor se pueda ejecutar de manera indefinida, sorteando una solicitud tras otra. A continuación, la dirección IP se enlaza con el socket y se realiza una verificación para ver si la llamada a *bind* tuvo éxito. El último paso en la inicialización es la llamada a *listen* para anunciar que el servidor está dispuesto a aceptar llamadas entrantes e indicar al sistema que almacene la cantidad de estas llamadas, que se especifica en QUEUE_SIZE, en caso de que lleguen más solicitudes mientras el servidor aún esté procesando la actual. Si la cola está llena y llegan solicitudes adicionales, se descartan en silencio.

En este punto el servidor entra a su ciclo principal, del cual nunca sale. La única forma de detenerlo es desde afuera. La llamada a *accept* bloquea el servidor hasta que algún cliente trata de establecer una conexión con él. Si la llamada a *accept* tiene éxito, se devuelve un descriptor de archivo que se puede utilizar para leer y escribir, de la misma forma en la que se pueden usar los descriptores de archivo para leer y escribir hacia/desde las canalizaciones. Sin embargo, a diferencia de las canalizaciones, que son unidireccionales, los sockets son bidireccionales, por lo que *sa* (el socket aceptado) se puede utilizar para leer de la conexión y también para escribir en ella. Un descriptor de archivo de canalización es para leer o escribir, pero no para ambas cosas.

Una vez que se establece la conexión, el servidor lee en ella el nombre del archivo. Si el nombre aún no está disponible, el servidor se bloquea y lo espera. Una vez que obtiene el nombre, el servidor abre el archivo y luego entra en un ciclo que alterna entre leer bloques del archivo y escribirlos en el socket hasta que el archivo se haya copiado por completo. A continuación, el servidor cierra el archivo y la conexión, y espera hasta que aparezca la siguiente conexión. Este ciclo se repite de manera indefinida.

Ahora analicemos el código del cliente. Para entender su funcionamiento es necesario comprender cómo se invoca. Suponiendo que se llama *cliente*, una llamada típica es:

```
cliente flits.cs.vu.nl /usr/tom/nombredearchivo >f
```

Esta llamada sólo funciona si el servidor ya se está ejecutando en *flits.cs.vu.nl,* si existe el archivo */usr/tom/nombredearchivo* y si el servidor tiene acceso de lectura a él. Si la llamada es exitosa, el archivo se transfiere a través de Internet y se escribe en *f*, después de lo cual finaliza el programa cliente. Puesto que el servidor continúa después de una transferencia, el cliente puede iniciarse una y otra vez para obtener otros archivos.

El código del cliente inicia con algunas inclusiones y declaraciones. La ejecución verifica primero si se invocó el código con el número correcto de argumentos (*argc* = 3 significa el nombre del programa más dos argumentos). Observe que *argv* [1] contiene el nombre del servidor (por ejemplo, *flits.cs.vu.nl*) y que *gethostbyname* lo convierte en una dirección IP. Esta función utiliza DNS para buscar el nombre. En el capítulo 7 estudiaremos el sistema DNS.

A continuación se crea e inicializa un socket. Después de esto, el cliente intenta establecer una conexión TCP con el servidor mediante *connect*. Si el servidor está activo y ejecutándose en la máquina especificada y enlazado a *SERVER_PORT*, y si está inactivo o tiene espacio en su cola *listen*, la conexión se establecerá (en algún momento dado). El cliente envía el nombre del archivo mediante esta conexión, para lo cual escribe en el socket. El número de bytes enviados es un byte mayor que el propio nombre, puesto que también hay que enviar el byte 0 de terminación del nombre para indicar al servidor en dónde termina dicho nombre.

Ahora el cliente entra en un ciclo, lee el archivo bloque por bloque desde el socket y lo copia a la salida estándar. Cuando termina, simplemente abandona la conexión.

El procedimiento *fatal* imprime un mensaje de error y termina. El servidor necesita el mismo procedimiento, sólo que lo omitimos debido a la falta de espacio en la página. Puesto que el cliente y el servidor se compilan por separado y, por lo general, se ejecutan en computadoras diferentes, no pueden compartir el código de *fatal*.

Estos dos programas (así como el material adicional relacionado con este libro) se pueden obtener del sitio web del libro

http://www.pearsonhighered.com/tanenbaum

Sólo como aclaración, este servidor no es lo último en tecnología de servidores. Su proceso de verificación de errores no es muy bueno y su proceso de reporte de errores es regular. Puesto que maneja todas las solicitudes estrictamente en forma secuencial (ya que cuenta con un solo hilo), su desempeño es pobre. Es evidente que nunca ha escuchado sobre la seguridad; además, utilizar sólo llamadas de sistema UNIX no es la manera de obtener independencia de la plataforma. También da por sentados algunos detalles que son técnicamente ilegales, como suponer que el nombre del archivo cabe en el búfer y se transmite instantáneamente. Pese a estas deficiencias, es un servidor de archivos de Internet funcional. En los ejercicios, invitamos al lector a mejorarlo. Para mayor información sobre la programación con sockets, consulte a Donahoo y Calvert (2008, 2009).

6.2 ELEMENTOS DE LOS PROTOCOLOS DE TRANSPORTE

El servicio de transporte se implementa mediante un **protocolo de transporte** entre las dos entidades de transporte. En ciertos aspectos, los protocolos de transporte se parecen a los protocolos de enlace de datos que estudiamos con detalle en el capítulo 3. Ambos se encargan del control de errores, la secuenciación y el control de flujo, entre otros aspectos.

Sin embargo, existen diferencias considerables entre los dos, las cuales se deben a disimilitudes importantes entre los entornos en que operan ambos protocolos, como se muestra en la figura 6-7. En la capa de enlace de datos, dos enrutadores se comunican de forma directa mediante un canal físico, ya sea cableado o inalámbrico, mientras que, en la capa de transporte, ese canal físico se sustituye por la red completa. Esta diferencia tiene muchas implicaciones importantes para los protocolos.

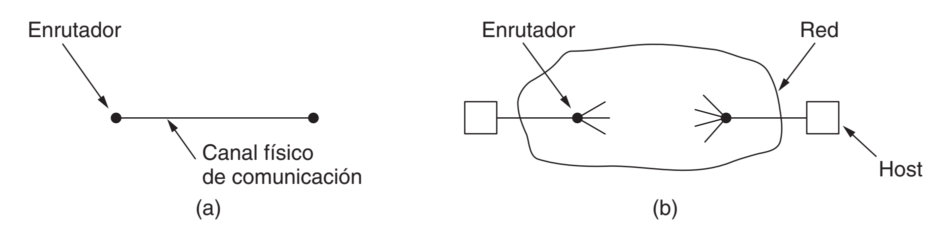


Figura 6-7. (a) Entorno de la capa de enlace de datos. (b) Entorno de la capa de transporte.

Por una parte, a través de los enlaces de punto a punto tales como los cables o la fibra óptica, por lo general no es necesario que un enrutador especifique con quién quiere comunicarse; cada línea de salida conduce de manera directa a un enrutador específico. En la capa de transporte se requiere el direccionamiento explícito de los destinos.

Por otro lado, el proceso de establecer una conexión a través del cable de la figura 6-7(a) es sencillo: el otro extremo siempre está ahí (a menos que se caiga, en cuyo caso no estará ahí). Sea como fuere, no hay mucho que hacer. Aún en enlaces inalámbricos el proceso no es muy diferente. Sólo basta enviar un mensaje para que llegue a todos los demás destinos. Si no se confirma la recepción del mensaje debido a un error, se puede reenviar. En la capa de transporte, el establecimiento inicial de la conexión es complicado, como veremos.

Otra diferencia (muy irritante) entre la capa de enlace de datos y la capa de transporte es la existencia potencial de capacidad de almacenamiento en la red. Cuando un enrutador envía un paquete a través de un enlace, éste puede llegar o perderse, pero no puede andar de un lado a otro durante un rato, esconderse en un rincón alejado del mundo y aparecer de repente después de otros paquetes que se hayan enviado mucho después. Si la red usa datagramas, los cuales se enrutan por separado en el interior, hay una probabilidad nada despreciable de que un paquete pueda tomar la ruta escénica, llegar tarde y fuera del orden esperado, o incluso que lleguen duplicados de ese paquete. Las consecuencias de la capacidad de la red de retrasar y duplicar paquetes algunas veces pueden ser desastrosas y puede requerir el uso de protocolos especiales para transportar la información de la manera correcta.

Una última diferencia entre las capas de enlace de datos y de transporte es de cantidad, más que de tipo. Se requieren búferes y control de flujo en ambas capas, pero la presencia de una cantidad de conexiones grande y variable en la capa de transporte, con un ancho de banda que fluctúa a medida que las conexiones compiten entre sí, puede requerir un enfoque distinto del que se usa en la capa de enlace de datos. Algunos de los protocolos que vimos en el capítulo 3 asignan una cantidad fija de búferes a cada línea de modo que, al llegar una trama, siempre hay un búfer disponible. En la capa de transporte, la gran cantidad de conexiones que se deben manejar, además de las variaciones en el ancho de banda que

puede recibir cada conexión, hacen menos atractiva la idea de dedicar muchos búferes a cada una. En las siguientes secciones examinaremos todos estos importantes temas, además de otros.

6.2.1 Direccionamiento

Cuando un proceso de aplicación (por ejemplo, un usuario) desea establecer una conexión con un proceso de aplicación remoto, debe especificar a cuál se conectará (el transporte sin conexión tiene el mismo problema: ¿a quién debe enviarse cada mensaje?). El método que se emplea por lo general es definir direcciones de transporte en las que los procesos puedan escuchar las solicitudes de conexión. En Internet, estos puntos terminales se denominan **puertos**. Usaremos el término genérico **TSAP** (**Punto de Acceso al Servicio de Transporte**, del inglés *Transport Service Access Point*) para indicar un punto terminal específico en la capa de transporte. Así, no es sorpresa que los puntos terminales análogos en la capa de red (es decir, direcciones de capa de red) se llamen **NSAP** (**Punto de Acceso al Servicio de Red**, del inglés *Network Service Access Points*). Las direcciones IP son ejemplos de NSAP.

En la figura 6-8 se ilustra la relación entre el NSAP, el TSAP y la conexión de transporte. Los procesos de aplicación, tanto clientes como servidores, se pueden enlazar por sí mismos a un TSAP para establecer una conexión a un TSAP remoto. Estas conexiones se realizan a través de puntos NSAP en cada host, como se muestra. El propósito de tener puntos TSAP es que, en algunas redes, cada computadora tiene un solo NSAP, por lo que se necesita alguna forma de diferenciar los múltiples puntos terminales de transporte que comparten ese punto NSAP.

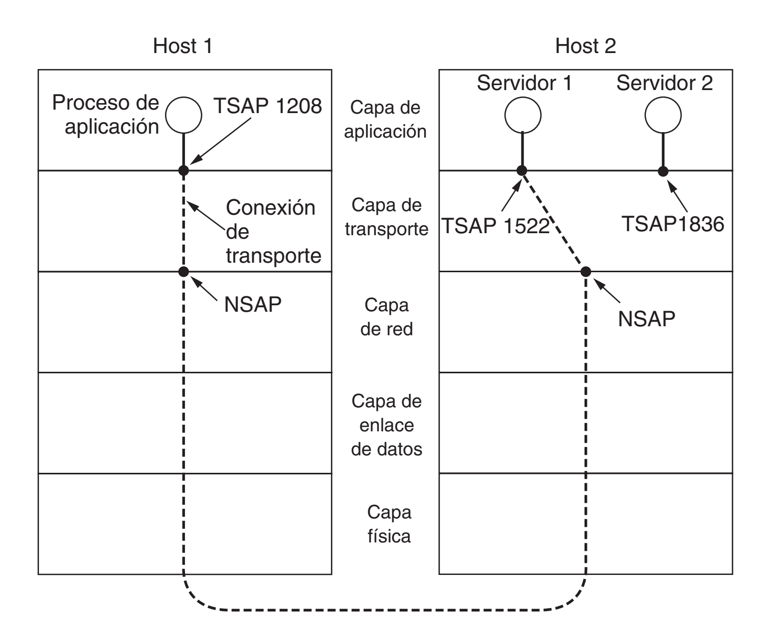


Figura 6-8. Los puntos TSAP y NSAP, junto con las conexiones de transporte.

El siguiente es un posible escenario para una conexión de transporte:

1. Un proceso servidor de correo se enlaza con el TSAP 1522 en el host 2 para esperar una llamada entrante. La manera en que un proceso se enlaza con un TSAP está fuera del modelo de red y depende por completo del sistema operativo local. Por ejemplo, se podría usar una llamada como nuestra LISTEN.

2. Un proceso de aplicación en el host 1 quiere enviar un mensaje de correo, por lo que se enlaza con el TSAP 1208 y emite una solicitud CONNECT. Esta solicitud especifica el TSAP 1208 en el host 1 como el origen y el TSAP 1522 en el host 2 como destino. En última instancia, esta acción provoca que se establezca una conexión de transporte entre el proceso de aplicación y el servidor.
3. El proceso de aplicación envía el mensaje de correo.
4. El servidor de correo responde para decir que entregará el mensaje.
5. Se libera la conexión de transporte.

Observe que en el host 2 podría haber otros servidores enlazados a otros puntos TSAP en espera de conexiones entrantes que lleguen a través del mismo NSAP.

El panorama anterior está muy bien, excepto que hemos ocultado un pequeño problema bajo la alfombra: ¿cómo sabe el proceso de usuario del host 1 que el servidor de correo está conectado al TSAP 1522? Una posibilidad es que el servidor de correo se haya estado enlazando con el TSAP 1522 durante años y que gradualmente todos los usuarios de la red hayan aprendido esto. En este modelo, los servicios tienen direcciones TSAP estables que se listan en archivos en lugares bien conocidos. Por ejemplo, el archivo */etc/services* de los sistemas UNIX, que lista cuáles servidores están enlazados de manera permanente a cuáles puertos, incluyendo el hecho de que el servidor de correo se encuentra en el puerto TCP 25.

Aunque las direcciones TSAP estables funcionan bien con una cantidad pequeña de servicios clave que nunca cambian (por ejemplo, el servidor web), en general, es común que los procesos de usuario deseen comunicarse con otros procesos de usuario que no tienen una dirección TSAP conocida por adelantado, o que sólo existan durante un tiempo corto.

Para manejar esta situación, podemos utilizar un esquema alternativo en el que existe un proceso especial llamado **asignador de puertos** (*portmapper*). Para encontrar la dirección TSAP correspondiente a un nombre de servicio específico, como “BitTorrent”, un usuario establece una conexión al asignador de puertos (que escucha en un TSAP bien conocido). Entonces, el usuario envía un mensaje en el que especifica el nombre del servicio y el asignador de puertos le regresa la dirección TSAP. A continuación, el usuario libera la conexión con el asignador de puertos y establece una nueva conexión con el servicio deseado.

En este modelo, cuando se crea un nuevo servicio, éste se debe registrar con el asignador de puertos para proporcionarle su nombre de servicio (por lo general, una cadena ASCII) y su TSAP. El asignador de puertos registra esta información en su base de datos interna, de manera que cuando empiecen a llegar las consultas más tarde, conozca las respuestas.

La función del asignador de puertos es análoga a la de un operador de asistencia de directorios en el sistema telefónico: provee una asociación de nombres con números. Al igual que en el sistema telefónico, es imprescindible que la dirección del TSAP bien conocido que utilice el asignador de puertos sea en realidad bien conocida. Si usted no conoce el número del operador de información, no puede llamarle para averiguarlo. Si cree que el número que marca para solicitar información es obvio, intente hacerlo en algún otro país cuando tenga oportunidad.

Muchos de los procesos servidor que pueden existir en una máquina sólo se utilizarán pocas veces. Es un desperdicio tenerlos a todos activos y escuchando en una dirección TSAP todo el día. En la figura 6-9 se muestra un esquema alternativo en forma simplificada. Este esquema se conoce como **protocolo de conexión inicial**. En vez de que cada uno de los servidores existentes escuche en un TSAP bien conocido, cada máquina que desea ofrecer servicios a usuarios remotos tiene un **servidor de procesos** especial, el cual actúa como proxy de los servidores que se usan menos. Este servidor se llama *inetd* en los sistemas UNIX, y escucha a un conjunto de puertos al mismo tiempo, en espera de una solicitud de conexión. Los usuarios potenciales de un servicio comienzan por emitir una solicitud CONNECT, en la que especifican la dirección TSAP del servicio que desean. Si no hay ningún servidor esperándolos, consiguen una conexión al servidor de procesos, como se muestra en la figura 6-9(a).

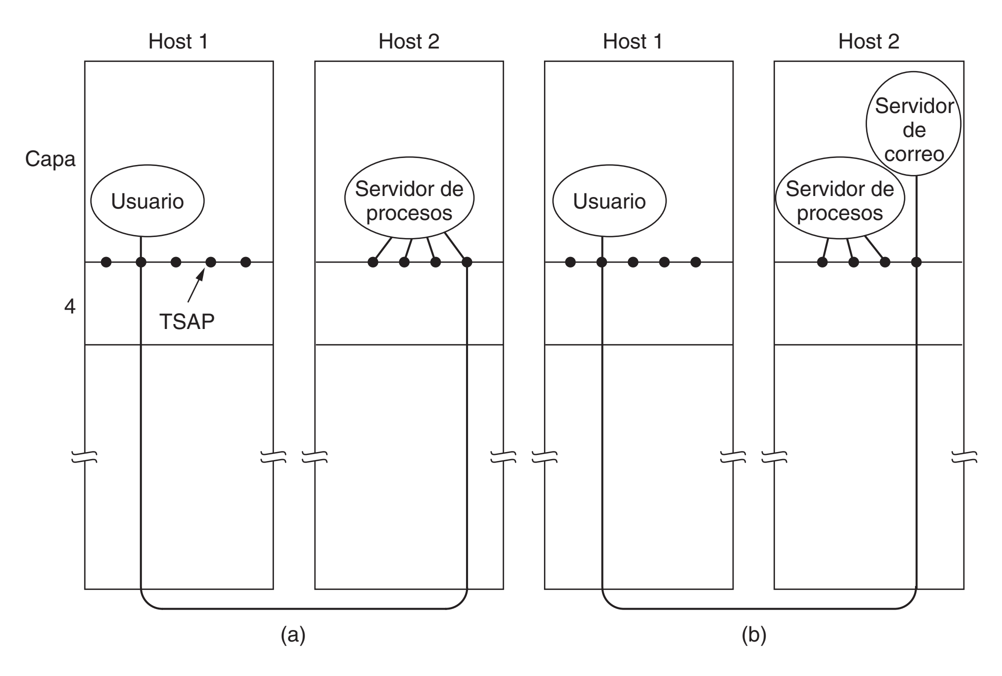


Figura 6-9. Cómo establece un proceso de usuario en el host 1 una conexión con un servidor de correo en el host 2, mediante un servidor de procesos.

Tras obtener la solicitud entrante, el servidor de procesos genera el servidor solicitado y le permite heredar la conexión existente con el usuario. El nuevo servidor hace el trabajo solicitado, mientras que el servidor de procesos regresa a escuchar nuevas solicitudes, como se muestra en la figura 6-9(b). Este método sólo se puede aplicar cuando los servidores se crean bajo demanda.

6.2.2 Establecimiento de una conexión

El proceso de establecer una conexión suena fácil, pero en realidad es sorprendentemente complicado. A primera vista parecería suficiente con que una entidad de transporte enviara tan sólo un segmento CONNECTION REQUEST al destino y esperara una respuesta CONNECTION ACCEPTED. El problema ocurre cuando la red puede perder, retrasar, corromper y duplicar paquetes. Este comportamiento causa complicaciones serias.

Imagine una red que está tan congestionada que las confirmaciones de recepción casi nunca regresan a tiempo, y cada paquete expira y se retransmite dos o tres veces. Suponga que la red usa datagramas en su interior y que cada paquete sigue una ruta diferente. Algunos de los paquetes podrían atorarse en un congestionamiento de tráfico dentro de la red y tardar mucho tiempo en llegar. Es decir, se pueden retardar en la red y reaparecer mucho después, cuando el emisor piense que se han perdido.

La peor pesadilla posible es la siguiente. Un usuario establece una conexión con un banco y envía mensajes para indicar al banco que transfiera una gran cantidad de dinero a la cuenta de una persona no del todo confiable. Por desgracia, los paquetes toman la ruta escénica hacia el destino y parten para explorar un rincón lejano de la red. Entonces el temporizador del emisor expira y envía todos los paquetes de nuevo. Esta vez los paquetes toman la ruta más corta y se entregan con rapidez, por lo que el emisor libera la conexión.

Por desgracia, en un momento dado el lote inicial de paquetes sale finalmente de su escondite y llega al destino en orden; le pide al banco que establezca una nueva conexión y que transfiera dinero (otra vez).

El banco no tiene manera de saber que estos paquetes son duplicados. Debe suponer que ésta es una segunda transacción independiente, y transfiere de nuevo el dinero.

Este escenario puede parecer poco probable o incluso inverosímil, pero el punto es el siguiente: los protocolos se deben diseñar de manera que sean correctos en todos los casos. Sólo es necesario implementar eficientemente los casos comunes para obtener un buen desempeño de la red, pero el protocolo debe ser capaz de lidiar con los casos no comunes sin quebrantarse. En caso de que no pueda, hemos construido una red poco confiable que puede fallar sin avisarnos cuando las condiciones se pongan difíciles.

Durante el resto de esta sección estudiaremos el problema de los duplicados con retardo, con un énfasis en los algoritmos para establecer conexiones de una manera confiable, de modo que no ocurran pesadillas como la anterior. El meollo del problema es que los duplicados con retardo se consideran paquetes nuevos. No podemos evitar que los paquetes se dupliquen y retrasen. Pero si esto ocurre, los paquetes deben ser rechazados como duplicados y no procesados como paquetes nuevos.

El problema se puede atacar de varias maneras, ninguna de las cuales es muy satisfactoria. Una es usar direcciones de transporte desechables. En este enfoque, cada vez que se requiere una dirección de transporte, se genera una nueva. Cuando una conexión es liberada, se descarta la dirección y no se vuelve a utilizar. Así, los paquetes duplicados con retardo nunca encuentran su camino hacia un proceso de transporte y no pueden hacer ningún daño. Sin embargo, esta estrategia dificulta la conexión con un proceso.

Otra posibilidad es dar a cada conexión un identificador único (es decir, un número de secuencia que se incremente con cada conexión establecida) elegido por la parte iniciadora y puesto en cada segmento, incluyendo el que solicita la conexión. Después de liberar cada conexión, cada entidad de transporte puede actualizar una tabla que liste conexiones obsoletas como pares (entidad de transporte de igual, identificador de conexión). Cada vez que entre una solicitud de conexión, se puede verificar con la tabla para saber si pertenece a una conexión previamente liberada.

Por desgracia, este esquema tiene una falla básica: requiere que cada entidad de transporte mantenga una cierta cantidad de información histórica durante un tiempo indefinido. Esta historia debe persistir tanto en la máquina de origen como en la de destino. De lo contrario, si una máquina falla y pierde su memoria, ya no sabrá qué identificadores de conexión ya han utilizado sus iguales.

Más bien, necesitamos un enfoque diferente para simplificar el problema. En lugar de permitir que los paquetes vivan eternamente dentro de la red, debemos idear un mecanismo para eliminar a los paquetes viejos que aún andan vagando por ahí. Con esta restricción, el problema se vuelve algo más manejable.

El tiempo de vida de un paquete puede restringirse a un máximo conocido mediante el uso de una (o más) de las siguientes técnicas:

1. Un diseño de red restringido.
2. Colocar un contador de saltos en cada paquete.
3. Marcar el tiempo en cada paquete.

La primera técnica incluye cualquier método que evite que los paquetes hagan ciclos, combinado con una manera de limitar el retardo, incluyendo la congestión a través de la trayectoria más larga posible (ahora conocida). Es difícil, dado que el alcance de las interredes puede variar, desde una sola ciudad hasta un alcance internacional. El segundo método consiste en inicializar el contador de saltos con un valor apropiado y decrementarlo cada vez que se reenvíe el paquete. El protocolo de red simplemente descarta cualquier paquete cuyo contador de saltos llega a cero. El tercer método requiere que cada paquete lleve la hora en la que fue creado, y que los enrutadores se pongan de acuerdo en descartar cualquier paquete que haya rebasado cierto tiempo predeterminado. Este último método requiere que los relojes de los enrutadores estén sincronizados, lo que no es una tarea fácil, además, en la práctica un contador de saltos es una aproximación suficientemente cercana a la edad.

En la práctica, necesitaremos garantizar no sólo que un paquete está eliminado, sino que todas sus confirmaciones de recepción también están eliminadas, por lo que ahora introduciremos un periodo *T*, que es un múltiplo pequeño del tiempo de vida máximo verdadero del paquete. El tiempo de vida máximo del paquete es una constante conservadora para una red; en Internet se toma de manera arbitraria como 120 segundos. El múltiplo depende del protocolo y simplemente tiene el efecto de hacer a *T* más largo. Si esperamos un tiempo de *T* segundos después de enviar un paquete, podemos estar seguros de que todos sus rastros ya han desaparecido, y que ni él ni sus confirmaciones de recepción aparecerán repentinamente de la nada para complicar el asunto.

Al limitar los tiempos de vida de los paquetes, es posible proponer una manera práctica y a prueba de errores para rechazar segmentos duplicados con retardo. El método descrito a continuación se debe a Tomlinson (1975); después Sunshine y Dalal (1978) lo refinaron. Las variantes de este método se utilizan mucho en la práctica, incluyendo en TCP.

La base del método es que el origen etiquete los segmentos con números de secuencia que no se vayan a reutilizar durante *T* segundos. El periodo *T* y la tasa de paquetes por segundo determinan el tamaño de los números de secuencia. De esta manera, sólo un paquete con un número de secuencia específico puede estar pendiente en cualquier momento dado. Aún puede haber duplicados de este paquete, en cuyo caso el destino debe descartarlos. Sin embargo, ya no se da el caso en que un duplicado con retardo de un paquete pueda vencer a un nuevo paquete con el mismo número de secuencia y que el destino lo acepte.

Para resolver el problema de una máquina que pierde toda la memoria acerca de su estado tras una falla, una posibilidad es exigir a las entidades de transporte que estén inactivas durante *T* segundos después de una recuperación. El periodo inactivo permitirá que todos los segmentos antiguos expiren, por lo que el emisor podrá empezar de nuevo con cualquier número de secuencia. Sin embargo, en una interred (interconexión de redes) compleja el periodo *T* puede ser grande, por lo que esta estrategia no es muy atractiva.

En cambio, Tomlinson propuso equipar cada host con un reloj. Los relojes de los distintos hosts no necesitan estar sincronizados. Se supone que cada reloj tiene la forma de un contador binario que se incrementa a sí mismo a intervalos uniformes. Además, la cantidad de bits del contador debe ser igual o mayor que la cantidad de bits en los números de secuencia. Por último, y lo más importante, se supone que el reloj continúa operando aunque el host falle.

Cuando se establece una conexión, los *k* bits de menor orden del reloj se usan como número de secuencia inicial de *k* bits. Por tanto, y a diferencia de los protocolos del capítulo 3, cada conexión comienza a numerar sus segmentos con un número de secuencia inicial diferente. El espacio de secuencia también debe ser lo bastante grande para que, para cuando los números de secuencia se reinicien, los segmentos antiguos con el mismo número de secuencia hayan desaparecido hace mucho tiempo. En la figura 6-10(a) se muestra esta relación lineal entre tiempo y números secuenciales iniciales. La región prohibida muestra los tiempos en donde los números de secuencia de los segmentos son ilegales previo a su uso. Si se envía cualquier segmento con un número de secuencia en esta región, se podría retrasar y hacerse pasar por un paquete distinto con el mismo número de secuencia que se emitirá un poco más tarde. Por ejemplo, si el host falla y se reinicia en la marca de tiempo de 70 segundos, utilizará los números de secuencia iniciales con base en el reloj para continuar a partir de donde se quedó; el host no empieza con un número de secuencia inferior en la región prohibida.

Una vez que ambas entidades de transporte han acordado el número de secuencia inicial, se puede usar cualquier protocolo de ventana deslizante para el control de flujo de datos. Este protocolo de ventana encontrará y descartará exactamente los paquetes duplicados después de que éstos hayan sido aceptados. En realidad, la curva de números de secuencia iniciales (que se indica mediante la línea gruesa) no es lineal, sino una escalera, ya que el reloj avanza en pasos discretos. Por sencillez, ignoraremos este detalle.

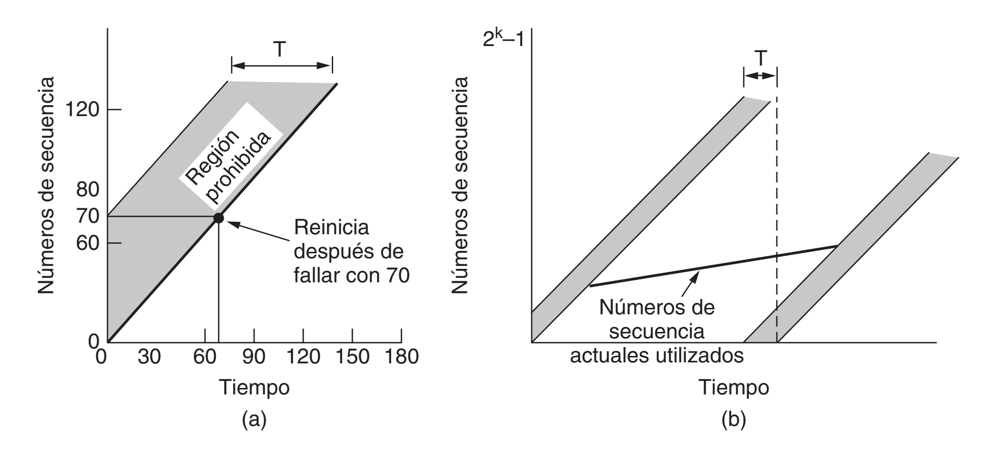


Figura 6-10. (a) Los segmentos no pueden entrar en la región prohibida. (b) El problema de resincronización.

Para mantener los números de secuencia fuera de la región prohibida, necesitamos tener cuidado con dos aspectos. Podemos meternos en problemas de dos maneras distintas. Si un host envía muchos datos con demasiada rapidez en una conexión recién abierta, el número de secuencia actual contra la curva de tiempo puede subir en forma más pronunciada que el número de secuencia inicial contra la curva de tiempo, lo cual provoca que el número de secuencia entre a la región prohibida. Para evitar que esto ocurra, la tasa máxima de datos en cualquier conexión es de un segmento por cada pulso de reloj. Esto significa que la entidad de transporte debe esperar hasta que el reloj emita un pulso antes de abrir una nueva conexión después de un reinicio por falla, no sea que el mismo número se utilice dos veces. Ambos puntos están a favor de un pulso corto de reloj (1 μseg o menor). Pero el reloj no puede pulsar demasiado rápido en relación con el número de secuencia. Para una tasa de reloj de C y un espacio de números de secuencia de tamaño S, la condición es que $S/C>T$ para que los números de secuencia no se reinicien con tanta rapidez.

Entrar a la región prohibida por la parte inferior al enviar con demasiada rapidez no es la única forma de meterse en problemas. De la figura 6-10(b) podemos ver que, a cualquier tasa de datos menor que la tasa de reloj, la curva de números de secuencia actuales utilizados vs el tiempo entrará en un momento dado a la región prohibida desde la izquierda, mientras los números de secuencia se reinician. Entre mayor sea la pendiente de los números de secuencia actuales, más se retardará este evento. Al evitar esta situación se limita el grado de lentitud con que los números de secuencia pueden avanzar en una conexión (o qué tanto pueden durar las conexiones).

El método basado en reloj resuelve el problema de no poder diferenciar los segmentos duplicados con retardo de los segmentos nuevos. Sin embargo, hay un inconveniente práctico en cuanto a su uso para establecer conexiones. Como por lo general no recordamos los números de secuencia de una conexión a otra en el destino, aún no tenemos forma de saber si un segmento CONNECTION REQUEST que contiene un número de secuencia inicial es un duplicado de una conexión reciente. Este inconveniente no existe durante una conexión, ya que el protocolo de la ventana deslizante sí recuerda el número de secuencia actual.

Para resolver este problema específico, Tomlinson (1975) desarrolló el **acuerdo de tres vías** (*three-way handshake*). Este protocolo de establecimiento implica que un igual verifique con el otro que la solicitud de conexión sea realmente actual. El procedimiento normal de establecimiento al iniciar el host 1 se muestra en la figura 6-11(a). El host 1 escoge un número de secuencia, x, y envía al host 2 un segmento CONNECTION REQUEST que contiene ese número. El host 2 responde con un segmento ACK para confirmar la recepción de x y anunciar su propio número de secuencia inicial, y. Por último, el host 1 confirma la recepción del número de secuencia inicial seleccionado por el host 2 en el primer segmento de datos que envía.

Ahora veamos la manera en que funciona el acuerdo de tres vías en presencia de segmentos de control duplicados con retardo. En la figura 6-11(b), el primer segmento es un CONNECTION REQUEST duplicado con retardo de una conexión antigua. Este segmento llega al host 2 sin el conocimiento del host 1. El host 2 reacciona a este segmento y envía al host 1 un segmento ACK, para solicitar en efecto la comprobación de que el host 1 haya tratado realmente de establecer una nueva conexión. Cuando el host 1 rechaza el intento del host 2 por establecer una conexión, el host 2 se da cuenta de que fue engañado por un duplicado con retardo y abandona la conexión. De esta manera, un duplicado con retardo no causa daño.

El peor caso ocurre cuando en la subred deambulan tanto un segmento CONNECTION REQUEST con retardo como un ACK. Este caso se muestra en la figura 6-11(c). Como en el ejemplo anterior, el host 2 recibe un CONNECTION REQUEST con retardo y lo contesta. En este momento es imprescindible tener en cuenta que el host 2 ha propuesto usar *y* como número de secuencia inicial para el tráfico del host 2 al host 1, sabiendo bien que no existen todavía segmentos que contengan el número de secuencia *y* ni confirmaciones de recepción de *y*. Cuando llega el segundo segmento con retardo al host 2, el hecho de

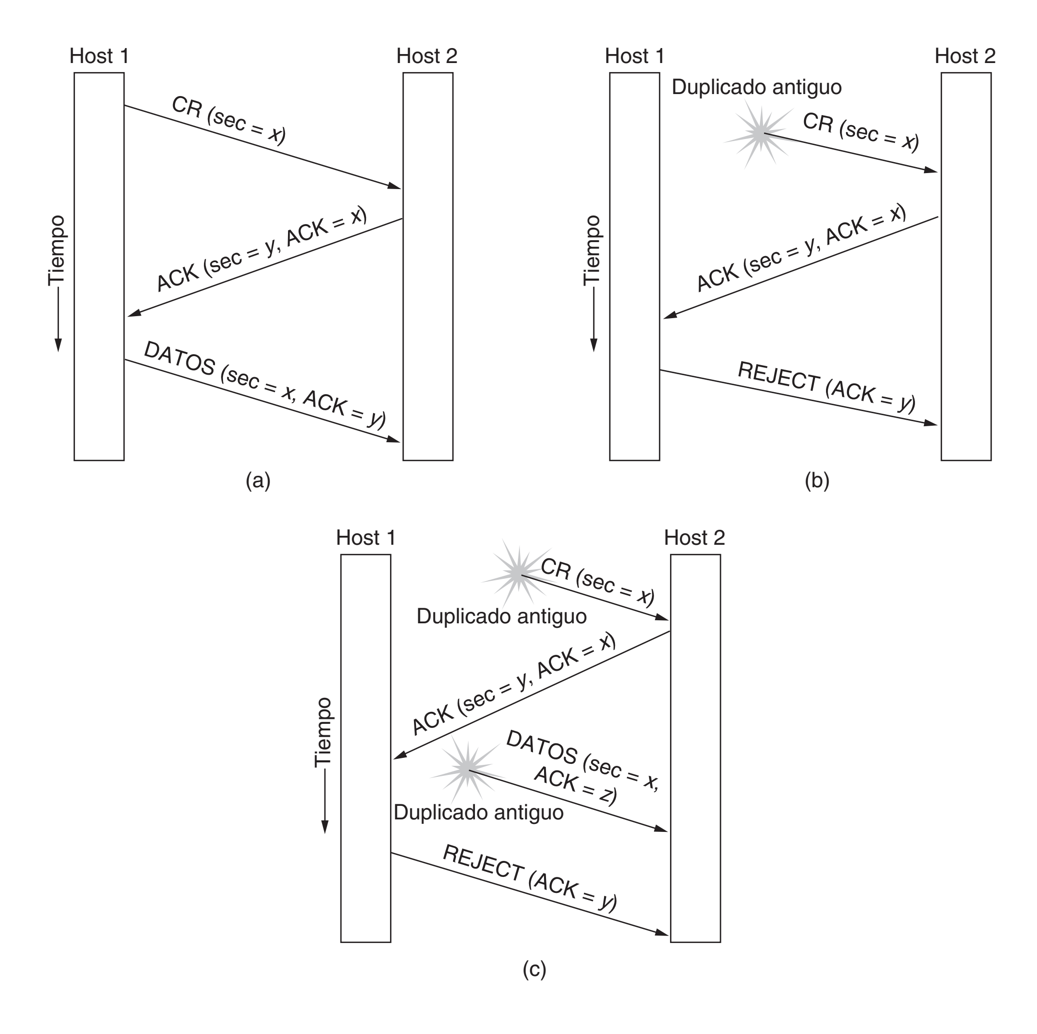


Figura 6-11. Tres escenarios del protocolo para establecer una conexión mediante un acuerdo de tres vías. CR significa CONNECTION REQUEST. (a) La operación normal. (b) Un CONNECTION REQUEST duplicado antiguo que aparece de la nada. (c) CONNECTION REQUEST duplicado y ACK duplicado.

que se confirmó la recepción de *z* en lugar de *y* indica al host 2 que éste también es un duplicado antiguo. Lo importante que debemos tener en cuenta aquí es que no hay una combinación de segmentos antiguos que puedan provocar la falla del protocolo y permitan establecer una conexión accidental cuando nadie la quiere.

TCP usa este acuerdo de tres vías para establecer las conexiones. Dentro de una conexión se utiliza una estampa de tiempo para extender el número de secuencia de 32 bits, de manera que no se reinicie en un intervalo menor al tiempo de vida máxima del paquete, incluso para las conexiones de gigabits por segundo. Este mecanismo es una corrección al TCP, la cual fue necesaria debido a que este protocolo se utilizaba en enlaces cada vez más rápidos. Se describe en el RFC 1323 y se llama **PAWS** (**Protección Contra el Reinicio de Números de Secuencia**, del inglés *Protection Against Wrapped Sequence Numbers*). Entre conexiones, para los números de secuencia y antes de que PAWS pudiera entrar en acción, TCP utilizaba originalmente el esquema basado en reloj que describimos antes. Sin embargo, se detectó una vulnerabilidad de seguridad: mediante el reloj, un atacante podía predecir con facilidad el siguiente número de secuencia inicial y enviar paquetes que engañaran al acuerdo de tres vías para establecer una conexión falsificada. Para cerrar este agujero, se utilizan números de secuencia inicial seudoaleatorios para las conexiones en la práctica. Sin embargo, aún es importante que no se repitan los números de secuencia iniciales durante un intervalo, incluso cuando parezcan aleatorios a quien los esté observando. En caso contrario, los duplicados con retardo pueden provocar un caos.

6.2.3 Liberación de una conexión

Es más fácil liberar una conexión que establecerla. No obstante, hay más obstáculos de los que uno podría imaginar. Como mencionamos antes, hay dos estilos para terminar una conexión: liberación asimétrica y liberación simétrica. La liberación asimétrica es la manera en que funciona el sistema telefónico: cuando una de las partes cuelga, se interrumpe la conexión. La liberación simétrica trata la conexión como dos conexiones unidireccionales distintas y requiere que cada una se libere por separado.

La liberación asimétrica es abrupta y puede provocar la pérdida de datos. Considere el escenario de la figura 6-12. Una vez que se establece la conexión, el host 1 envía un segmento que llega en forma apropiada al host 2. A continuación, el host 1 envía otro segmento. Por desgracia, el host 2 emite un DISCONNECT antes de que llegue el segundo segmento. El resultado es que se libera la conexión y se pierden los datos.

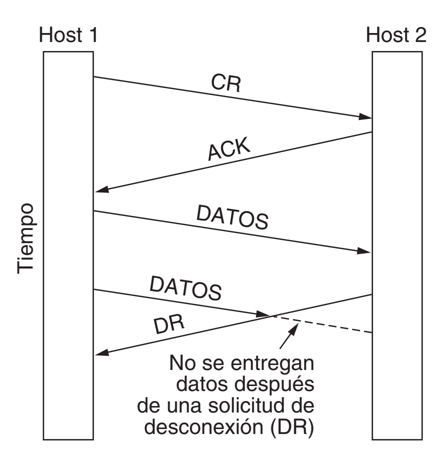


Figura 6-12. Desconexión abrupta con pérdida de datos.

Es obvio que se requiere un protocolo de liberación más sofisticado para evitar la pérdida de los datos. Una posibilidad es usar la liberación simétrica, en la que cada dirección se libera en forma independiente de la otra. Aquí, un host puede continuar recibiendo datos, aun después de haber enviado un segmento DISCONNECT.

La liberación simétrica es ideal cuando cada proceso tiene una cantidad fija de datos por enviar y sabe con certeza cuándo los ha enviado. En otras situaciones, el proceso de determinar si se ha efectuado o no todo el trabajo y si debe terminar o no la conexión no es tan obvio. Podríamos pensar en un protocolo en el que el host 1 diga: "Ya terminé. ¿Terminaste también?" Si el host 2 responde: "Ya terminé también. Adiós", la conexión se puede liberar sin problemas.

Por desgracia, este protocolo no siempre funciona. Hay un problema famoso que tiene que ver con ese asunto. Se conoce como el **problema de los dos ejércitos**. Imagine que un ejército blanco está acampado en un valle, como se muestra en la figura 6-13. En los dos cerros que rodean al valle hay ejércitos azules. El ejército blanco es más grande que cualquiera de los dos ejércitos azules por separado, pero juntos éstos son más grandes que el ejército blanco. Si cualquiera de los dos ejércitos azules ataca por su cuenta, será derrotado, pero si los dos atacan a la vez obtendrán la victoria.

Los ejércitos azules quieren sincronizar sus ataques. Sin embargo, su único medio de comunicación es el envío de mensajeros a pie a través del valle, donde podrían ser capturados y se perdería el mensaje (es decir, tienen que usar un canal de comunicación no confiable). La pregunta es: ¿existe un protocolo que permita que los ejércitos azules ganen?

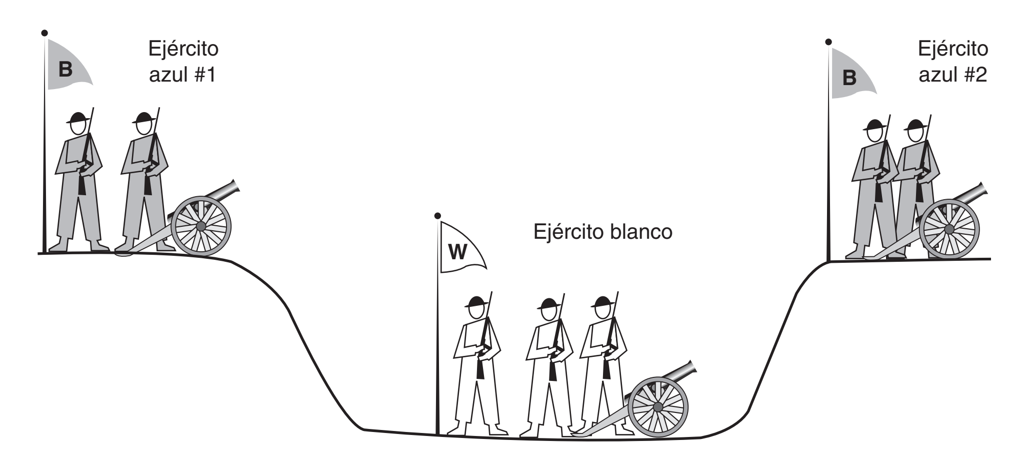


Figura 6-13. El problema de los dos ejércitos.

Supongamos que el comandante del ejército azul #1 envía un mensaje que dice: "Propongo que ataquemos al amanecer del 29 de marzo. ¿Qué les parece?" Ahora supongamos que llega el mensaje y que el comandante del ejército azul #2 está de acuerdo, y que su respuesta llega con seguridad al ejército azul #1. ¿Ocurrirá el ataque? Es probable que no, porque el comandante #2 no sabe si su respuesta llegó. Si no llegó, el ejército azul #1 no atacará, y sería tonto de su parte emprender el combate.

Mejoremos ahora el protocolo para convertirlo en un acuerdo de tres vías. El iniciador de la propuesta original debe confirmar la recepción de la respuesta. Suponiendo que no se pierden mensajes, el ejército azul #2 recibirá la confirmación de recepción, pero ahora el que dudará será el comandante del ejército azul #1. A fin de cuentas, no sabe si ha llegado su confirmación de recepción y, si no llegó, sabe que el ejército #2 no atacará. Podríamos probar ahora un protocolo de acuerdo de cuatro vías, pero tampoco ayudaría.

De hecho, podemos demostrar que no existe un protocolo que funcione. Supongamos que existiera algún protocolo. O el último mensaje del protocolo es esencial, o no lo es. Si no lo es, podemos eliminarlo (así como los demás mensajes no esenciales) hasta que quede un protocolo en el que todos los mensajes sean esenciales. ¿Qué ocurre si el mensaje final no pasa? Acabamos de decir que es esencial, por lo que, si se pierde, el ataque no ocurrirá. Dado que el emisor del mensaje final nunca puede estar seguro de su llegada, no se arriesgará a atacar. Peor aún, el otro ejército azul sabe esto, por lo que tampoco atacará.

Para ver la relevancia del problema de los dos ejércitos en relación con la liberación de conexiones, simplemente sustituya "atacar" por "desconectar". Si ninguna de las partes está preparada para desconectarse hasta estar convencida de que la otra está preparada para desconectarse también, nunca ocurrirá la desconexión.

En la práctica podemos evitar este dilema al eludir la necesidad de un acuerdo y pasar el problema al usuario de transporte, de modo que cada lado pueda decidir por su cuenta si se completó o no la comunicación. Éste es un problema más fácil de resolver. En la figura 6-14 se ilustran cuatro escenarios de liberación mediante el uso de un acuerdo de tres vías. Aunque este protocolo no es infalible, es adecuado en la mayoría de los casos.

En la figura 6-14(a) vemos el caso normal en el que uno de los usuarios envía un segmento DR de solicitud de desconexión (DISCONNECTION REQUEST) con el fin de iniciar la liberación de una conexión. Al llegar, el receptor devuelve también un segmento DR e inicia un temporizador, por si acaso se pierde su DR. Cuando este DR llega, el emisor original envía de regreso un segmento ACK y libera la conexión. Finalmente, cuando llega el segmento ACK, el receptor también libera la conexión. Liberar una conexión significa que la entidad de transporte remueve la información sobre la conexión de su tabla de conexiones abiertas y avisa de alguna manera al dueño de la conexión (el usuario de transporte). Esta acción es diferente a aquélla en la que el usuario de transporte emite una primitiva DISCONNECT.

Si se pierde el último segmento ACK, como se muestra en la figura 6-14(b), el temporizador salva la situación. Al expirar el temporizador, la conexión se libera de todos modos.

Ahora consideremos el caso en el que se pierde el segundo DR. El usuario que inicia la desconexión no recibirá la respuesta esperada, su temporizador expirará y todo comenzará de nuevo. En la figura 6-14(c) vemos la manera en que esto funciona, suponiendo que la segunda vez no se pierden segmentos, y que todos se entregan correctamente y a tiempo.

Nuestro último escenario, la figura 6-14(d), es el mismo que en la figura 6-14(c), excepto que ahora suponemos que todos los intentos repetidos de retransmitir el segmento DR también fallan debido a los segmentos perdidos. Después de N reintentos, el emisor simplemente se da por vencido y libera la conexión. Mientras tanto, expira el temporizador del receptor y también se sale.

Aunque por lo general basta con este protocolo, en teoría puede fallar si se pierden el DR inicial y N retransmisiones. El emisor se dará por vencido y liberará la conexión, pero el otro lado no sabrá nada sobre los intentos de desconexión y seguirá plenamente activo. Esta situación provoca una conexión semiabierta.

Pudimos haber evitado este problema al impedir que el emisor se diera por vencido después de N reintentos y obligarlo a seguir insistiendo hasta recibir una respuesta. No obstante, si permitimos que expire el temporizador en el otro lado, entonces el emisor continuará por siempre, pues nunca llegará una respuesta. Si no permitimos que expire el temporizador en el lado receptor, el protocolo queda suspendido en la figura 6-14(d).

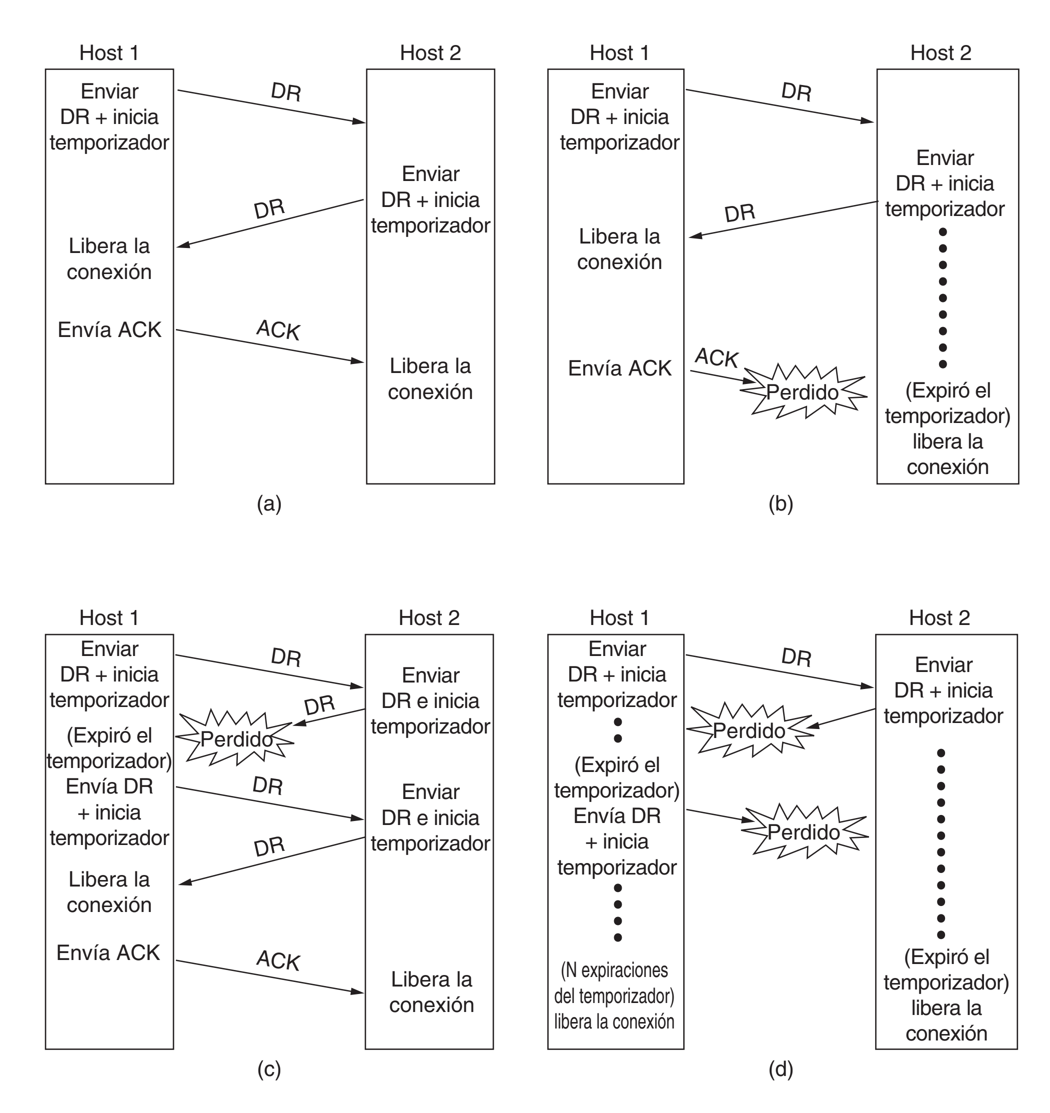


Figura 6-14. Cuatro escenarios de un protocolo para liberar una conexión. (a) Caso normal del acuerdo de tres vías. (b) Pérdida del último ACK. (c) Respuesta perdida. (d) Respuesta perdida y pérdida de los segmentos DR subsecuentes.

Una manera de eliminar las conexiones semiabiertas es tener una regla que diga que, si no han llegado segmentos durante ciertos segundos, se libera automáticamente la conexión. De esta manera, si un lado llega a desconectarse, el otro lado detectará la falta de actividad y también se desconectará. Esta regla también se encarga del caso donde se rompe la conexión (porque la red ya no puede entregar paquetes entre los hosts) sin que ninguno de los extremos se desconecte. Por supuesto que si se pone en práctica esta regla, es necesario que cada entidad de transporte tenga un temporizador que se detenga y se reinicie cada vez que se envíe un segmento. Si expira este temporizador se transmite un segmento ficticio, simplemente para evitar que el otro lado se desconecte. Por otra parte, si se usa la regla de desconexión automática y se pierden demasiados segmentos ficticios seguidos en una conexión que de otro modo estaría en reposo, primero se desconectará de forma automática un lado y después el otro.

No insistiremos más en este punto, pero ya debe quedar claro que liberar una conexión sin pérdida de datos no es ni remotamente tan simple como parece a primera instancia. Aquí la lección es que el usuario de transporte debe estar involucrado a la hora de decidir cuándo desconectarse; el problema no se puede resolver tan sólo mediante las entidades de transporte. Para ver la importancia de la aplicación, considere que mientras TCP realiza por lo general un cierre simétrico (en donde cada lado cierra de manera independiente su mitad de la conexión con un paquete FIN cuando termina de enviar sus datos), muchos servidores web envían al cliente un paquete RST que provoca un cierre repentino de la conexión, lo cual es más parecido a un cierre asimétrico. Esto funciona sólo porque el servidor web conoce el patrón del intercambio de datos. Primero recibe una solicitud del cliente, que incluye todos los datos que enviará ese cliente, y después envía una respuesta al cliente. Cuando el servidor web termina con su respuesta, se han enviado todos los datos en ambas direcciones. El servidor puede enviar una advertencia al cliente y cerrar en forma repentina la conexión. Si el cliente recibe esta advertencia, liberará su estado de conexión en ese momento. Si no recibe la advertencia, en algún momento detectará que el servidor ya no se está comunicando con él y liberará el estado de la conexión. En cualquiera de los casos, los datos se transfieren con éxito.

6.2.4 Control de errores y almacenamiento en búfer

Ya que examinamos cómo establecer y liberar conexiones con cierto detalle, veamos ahora la manera en que se manejan las conexiones mientras están en uso. Las cuestiones clave son el control de errores y el control de flujo. El control de errores consiste en asegurar que los datos se entreguen con el nivel deseado de confiabilidad, por lo general que todos los datos se entreguen sin errores. El control de flujo consiste en evitar que un transmisor rápido sature a un receptor lento.

Ya vimos ambas cuestiones antes, cuando estudiamos la capa de enlace de datos. Las soluciones que se utilizan en la capa de transporte son los mismos mecanismos que estudiamos en el capítulo 3. A continuación veremos una muy breve recapitulación:

1. Una trama transporta un código de detección de errores (por ejemplo, CRC o suma de verificación), el cual se utiliza para comprobar que la información se haya recibido de manera correcta.
2. Una trama transporta un número de secuencia para identificarse a sí misma; el emisor la retransmite hasta que reciba una confirmación de recepción exitosa por parte del receptor. A esto se le conoce como **ARQ** (**Solicitud Automática de Repetición**, del inglés *Automatic Repeat reQuest*).
3. Hay un número máximo de tramas pendientes que el emisor permitirá en un momento dado, y se detendrá si el receptor no envía confirmaciones de recepción de las tramas con la rapidez suficiente. Si este máximo es de un paquete, el protocolo se denomina **parada y espera**. Las ventanas más grandes permiten canalizaciones y mejoran el desempeño en enlaces largos y rápidos.
4. El protocolo de la **ventana deslizante** combina estas características y también se utiliza para soportar la transferencia de datos bidireccional.

Dado que estos mecanismos se utilizan en tramas de la capa de enlace, es natural preguntarse por qué se utilizarían también en segmentos de la capa de transporte. Sin embargo, en la práctica hay una pequeña duplicación entre las capas de enlace y de transporte. A pesar de que se utilizan los mismos mecanismos, existen diferencias en cuanto a función y grado.

Para una diferencia en función, considere la detección de errores. La suma de verificación de la capa de enlace protege una trama mientras atraviesa un solo enlace. La suma de verificación de la capa de

transporte protege un segmento mientras atraviesa una trayectoria de red completa. Es una verificación de punto a punto; que no es lo mismo que realizar una verificación en cada enlace. Saltzer y colaboradores (1984) describen una situación en que los paquetes se corrompieron dentro de un enrutador. Las sumas de verificación de la capa de enlace protegían a los paquetes sólo mientras viajaban a través de un enlace, no mientras estaban dentro del enrutador. Por ende, los paquetes se entregaron en forma incorrecta, incluso cuando eran correctos de acuerdo con las verificaciones en cada enlace.

Éste y otros ejemplos condujeron a Saltzer y colaboradores a que articularan el **argumento punto a punto**. De acuerdo con este argumento, la verificación de la capa de transporte que se efectúa de un punto a otro es esencial para la precisión, y aunque las verificaciones de la capa de enlace no son esenciales, sí son valiosas para mejorar el desempeño (ya que sin ellas se puede enviar innecesariamente un paquete corrupto a través de toda la ruta).

Como una diferencia de grado, considere las retransmisiones y el protocolo de ventana deslizante. Con excepción de los enlaces de satélite, la mayoría de los enlaces inalámbricos sólo pueden tener una trama pendiente del emisor en cualquier momento dado. Esto es, el producto de ancho de banda-retardo para el enlace es tan pequeño que ni siquiera se puede almacenar una trama completa dentro del enlace. En este caso, un tamaño de ventana pequeño es suficiente para un buen desempeño. Por ejemplo, 802.11 usa un protocolo de parada y espera, en donde transmite o retransmite cada trama y espera a recibir una confirmación de recepción antes de pasar a la siguiente trama. Un tamaño de ventana más grande aumentaría la complejidad sin mejorar el desempeño. Para los enlaces cableados y de fibra óptica, como Ethernet (conmutada) o las redes troncales de ISP, la tasa de errores es tan baja que se pueden omitir las retransmisiones de la capa de enlace, porque las retransmisiones punto a punto repararán la pérdida de tramas residuales.

Por otro lado, muchas conexiones TCP tienen un producto de ancho de banda-retardo mucho mayor que un solo segmento. Considere una conexión que envía datos a través de Estados Unidos a 1 Mbps y con un tiempo de ida y vuelta de 100 mseg. Incluso para esta conexión lenta se almacenarán 200 Kbits de datos en el receptor, en el tiempo requerido para enviar un segmento y recibir una confirmación de recepción. Para estos casos se debe usar una ventana deslizante grande. El protocolo de parada y espera paralizaría el desempeño. En nuestro ejemplo, limitaría el desempeño a un segmento cada 200 mseg, o 5 segmentos/seg sin importar qué tan rápida sea en realidad la red.

Dado que, por lo general, los protocolos de transporte usan ventanas deslizantes más grandes, analizaremos con más cuidado la cuestión de colocar los datos en un búfer. Como un host puede tener muchas conexiones, cada una de las cuales se trata por separado, puede requerir una cantidad considerable de búferes para las ventanas deslizantes. Se necesitan búferes tanto en el emisor como en el receptor. Sin duda, se requieren en el emisor para contener todos los segmentos transmitidos, para los cuales todavía no se ha enviado una confirmación de recepción. Se precisan ahí debido a que estos segmentos se pueden perder y tal vez necesiten retransmitirse.

Sin embargo, como el emisor está usando búferes, tal vez el receptor pueda o no dedicar búferes específicos a conexiones específicas, según lo considere apropiado. Por ejemplo, el receptor puede mantener un solo grupo de búferes compartido por todas las conexiones. Cuando entre un segmento, se hará un intento por adquirir un nuevo búfer en forma dinámica. Si hay uno disponible, se acepta el segmento; en caso contrario, se descarta. Como el emisor está preparado para retransmitir los segmentos perdidos por la red, no hay daño permanente al permitir que el receptor descarte segmentos, aunque se desperdician algunos recursos. El emisor simplemente sigue intentando hasta que recibe una confirmación de recepción.

La mejor solución de compromiso entre usar búferes en el origen o usarlos en el destino depende del tipo de tráfico transmitido por la conexión. Para el tráfico en ráfagas con bajo ancho de banda, como el que produce una terminal interactiva, es razonable no dedicar ningún búfer, sino adquirirlos en forma dinámica en ambos extremos, confiando en el uso de búferes en el emisor si hay que descartar segmentos ocasionalmente. Por otra parte, para la transferencia de archivos y demás tráfico de alto ancho de banda,

es mejor si el receptor dedica una ventana completa de búferes para permitir que los datos fluyan a la máxima velocidad. Ésta es la estrategia que utiliza TCP.

Todavía queda pendiente la cuestión de cómo organizar el grupo de búferes. Si la mayoría de los segmentos tienen el mismo tamaño aproximado, es natural organizar los búferes como un grupo de búferes de tamaño idéntico, con un segmento por búfer, como en la figura 6-15(a). Pero si hay una variación amplia en cuanto al tamaño de los segmentos, desde solicitudes cortas de páginas web hasta paquetes extensos en transferencias de archivos de igual a igual, un grupo de búferes de tamaño fijo presenta problemas. Si el tamaño de búfer se selecciona de manera que sea igual al segmento más grande, se desperdiciará espacio cada vez que llegue un segmento corto. Si el tamaño se selecciona de manera que sea menor al tamaño máximo de segmento, se requerirán varios búferes para los segmentos largos, con la complejidad inherente.

Otra forma de enfrentar el problema del tamaño de los búferes es usar búferes de tamaño variable, como en la figura 6-15(b). La ventaja aquí es un mejor uso de la memoria, al costo de una administración de búferes más complicada. Una tercera posibilidad es dedicar un solo búfer circular grande por conexión, como en la figura 6-15(c). Este sistema es simple y elegante, además de que no depende de los tamaños de los segmentos, pero hace buen uso de la memoria sólo cuando todas las conexiones están muy cargadas.

A medida que se abren y cierran conexiones y cambia el patrón del tráfico, el emisor y el receptor necesitan ajustar en forma dinámica sus asignaciones de búferes. En consecuencia, el protocolo de transporte debe permitir que un host emisor solicite espacio de búfer en el otro extremo. Se podrían asignar búferes por conexión o en forma colectiva, para todas las conexiones entre los dos hosts. De manera alternativa, al conocer su situación de uso de búferes (pero sin conocer el tráfico ofrecido), el receptor podría decir al emisor "Reservé *X* búferes para ti". Si el número de conexiones abiertas aumentara, tal vez sería necesario reducir una asignación, de modo que el protocolo debe tener en cuenta esta posibilidad.

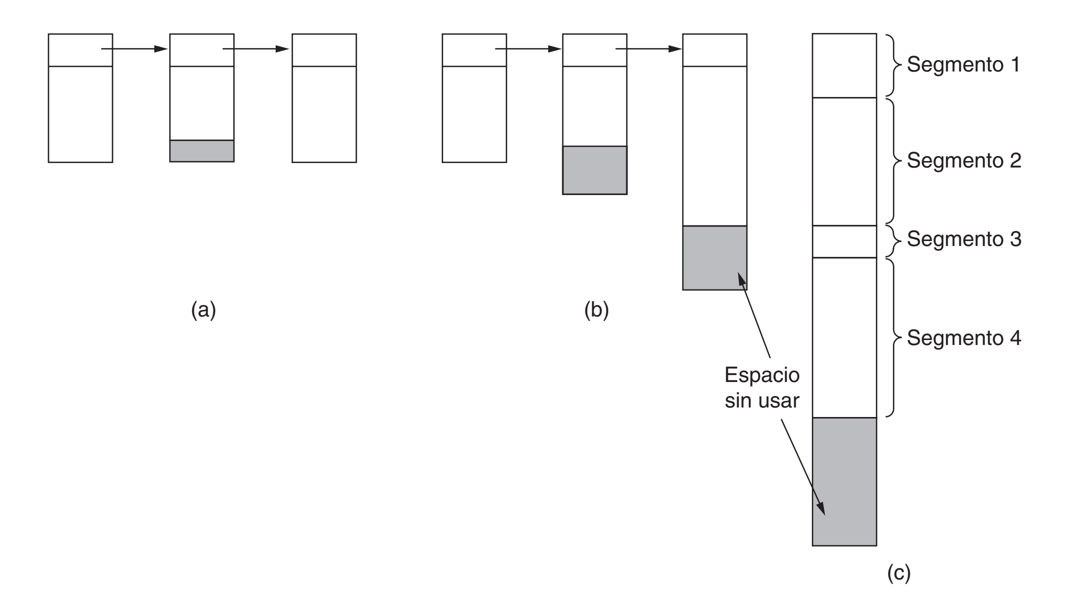


Figura 6-15. (a) Búferes encadenados de tamaño fijo. (b) Búferes encadenados de tamaño variable. (c) Un gran búfer circular por conexión.

Una manera razonable y general de administrar la asignación dinámica de búferes es desacoplar los búferes de las confirmaciones de recepción, en contraste a los protocolos de ventana deslizante del capítulo 3. En efecto, administración dinámica de búfer significa una ventana de tamaño variable. Al principio

el emisor solicita cierto número de búferes, con base en sus necesidades esperadas. Después el receptor otorga tantos de éstos como pueda. Cada vez que el emisor transmite un segmento debe disminuir su asignación, y cuando ésta llegue a cero debe detenerse por completo. El receptor superpone por separado las confirmaciones de recepción y las asignaciones de búfer en el tráfico inverso. TCP usa este esquema y transmite las asignaciones de búfer en un campo de encabezado llamado *Tamaño de ventana*.

La figura 6-16 muestra un ejemplo de la manera en que podría trabajar la administración dinámica de ventanas en una red de datagramas con números de secuencia de 4 bits. En este ejemplo, los datos fluyen en segmentos del host *A* al host *B*; las confirmaciones de recepción y las asignaciones de búferes fluyen en segmentos en dirección opuesta. En un principio *A* quiere ocho búferes, pero se le otorgan solamente cuatro. Después envía tres segmentos, de los cuales se pierde el tercero. El segmento 6 confirma la recepción de todos los segmentos hasta el número de secuencia 1, incluyéndolo, con lo cual permite que *A* libere esos búferes; además informa a *A* que tiene permiso de enviar tres segmentos más que empiecen después de 1 (es decir, los segmentos 2, 3 y 4). *A* sabe que ya ha enviado el número 2, por lo que piensa que debe enviar los segmentos 3 y 4, y procede a hacerlo. En este punto se bloquea y debe esperar una nueva asignación de búfer. Sin embargo, las retransmisiones inducidas por expiraciones del temporizador (línea 9) pueden ocurrir durante el bloqueo, pues usan búferes que ya se han asignado. En la línea 10, *B* confirma la recepción de todos los segmentos hasta el 4, inclusive, pero se niega a permitir que *A* continúe. Tal situación es imposible con los protocolos de ventana fija del capítulo 3. El siguiente segmento de *B* a *A* asigna otro búfer y permite a *A* continuar. Esto ocurrirá cuando *B* tenga espacio de búfer, quizá debido a que el usuario de transporte aceptó más segmentos de datos.

	A	Mensaje	B	Comentarios
1	→	< solicita 8 búferes >	→	A quiere 8 búferes.
2	←	<ack = 15, buf = 4>	←	B sólo otorga los mensajes 0 a 3.
3	→	<seq = 0, data = m0>	→	A tiene 3 búferes libres ahora.
4	→	<seq = 1, data = m1>	→	A tiene 2 búferes libres ahora.
5	→	<seq = 2, data = m2>	• • •	Se perdió el mensaje, pero A piensa que le queda 1 libre.
6	←	<ack = 1, buf = 3>	←	B confirma la recepción de 0 y 1, permite 2-4.
7	→	<seq = 3, data = m3>	→	A tiene un búfer libre.
8	→	<seq = 4, data = m4>	→	A tiene 0 búferes libres y debe detenerse.
9	→	<seq = 2, data = m2>	→	El temporizador de A expira y retransmite.
10	←	<ack = 4, buf = 0>	←	Todo confirmado, pero A sigue bloqueado.
11	←	<ack = 4, buf = 1>	←	A puede enviar el 5 ahora.
12	←	<ack = 4, buf = 2>	←	B encontró un nuevo búfer en alguna parte.
13	→	<seq = 5, data = m5>	→	A tiene 1 búfer libre.
14	→	<seq = 6, data = m6>	→	A está bloqueado nuevamente.
15	←	<ack = 6, buf = 0>	←	A sigue bloqueado.
16	• • •	<ack = 6, buf = 4>	←	Interbloqueo potencial.

Figura 6-16. Asignación dinámica de búferes. Las flechas muestran la dirección de la transmisión. Los puntos suspensivos (…) indican un segmento perdido.

Pueden surgir problemas con los esquemas de asignación de búferes de este tipo en las redes de datagramas si hay posibilidad de perder segmentos de control (que casi siempre es así). Observe la línea 16. *B* ha asignado ahora más búferes a *A*, pero el segmento de asignación se perdió. Dado que los segmentos de control no están en secuencia ni ha expirado su temporizador, *A* se encuentra en interbloqueo. Para evitar esta situación, cada host debe enviar periódicamente segmentos de control con la confirmación de

recepción y el estado de búferes de cada conexión. De esta manera se puede interrumpir el interbloqueo, tarde o temprano.

Hasta ahora hemos supuesto tácitamente que el único límite impuesto sobre la tasa de datos del emisor es la cantidad de espacio de búfer disponible en el receptor. A menudo éste no es el caso. Hace algún tiempo la memoria era costosa, pero en la actualidad los precios han disminuido mucho. Los hosts pueden estar equipados con tanta memoria que la falta de búferes deja de ser un problema, incluso para conexiones de área amplia. Desde luego que esto depende de que el tamaño del búfer se establezca con la capacidad suficiente, lo cual no siempre ha sido así para TCP (Zhang y colaboradores, 2002).

Si el espacio de búfer ya no limita el flujo máximo, aparecerá otro cuello de botella: la capacidad de transporte de la red. Si enrutadores adyacentes pueden intercambiar cuando mucho x paquetes/seg y hay k trayectorias separadas entre un par de hosts, no hay manera de que esos hosts puedan intercambiar más de kx segmentos/seg, sin importar la cantidad de espacio de búfer disponible en cada terminal. Si el emisor presiona demasiado (es decir, envía más de kx segmentos/seg), la red se congestionará pues será incapaz de entregar los segmentos a la velocidad con que llegan.

Lo que se necesita es un mecanismo que limite las transmisiones del emisor con base en la capacidad de transporte de la red, en lugar de basarse en la capacidad de almacenamiento en búfer del receptor. Belsnes (1975) propuso el uso de un esquema de control de flujo de ventana deslizante, en el que el emisor ajusta en forma dinámica el tamaño de la ventana para igualarla a la capacidad de transporte de la red. Esto significa que una ventana deslizante dinámica puede implementar tanto el control de flujo como el de congestión. Si la red puede manejar c segmentos/seg y el tiempo de ida y vuelta (incluyendo transmisión, propagación, encolamiento, procesamiento en el receptor y devolución de la confirmación de recepción) es de r, entonces la ventana del emisor debe ser cr. Con una ventana de este tamaño, el emisor normalmente opera con el canal a su máxima capacidad. Cualquier pequeña disminución en el desempeño de la red causará que se bloquee. Como la capacidad de red disponible para cualquier flujo dado varía con el tiempo, hay que ajustar el tamaño de ventana con frecuencia para rastrear los cambios en la capacidad de transporte. Como veremos más adelante, TCP usa un esquema similar.

6.2.5 Multiplexión

La multiplexión o la compartición de varias conversaciones a través de conexiones, circuitos virtuales y enlaces físicos, desempeña un papel importante en varias capas de la arquitectura de red. En la capa de transporte puede surgir la necesidad de usar la multiplexión de varias formas. Por ejemplo, si sólo hay una dirección de red disponible en un host, todas las conexiones de transporte de esa máquina tendrán que utilizarla. Cuando llega un segmento, se necesita algún mecanismo para saber a cuál proceso asignarlo. Esta situación, conocida como **multiplexión**, se muestra en la figura 6-17(a). En esta figura, cuatro conexiones de transporte distintas utilizan la misma conexión de red (por ejemplo, dirección IP) al host remoto.

La multiplexión también puede ser útil en la capa de transporte por otra razón. Por ejemplo, supongamos que un host cuenta con varias trayectorias de red que puede utilizar. Si un usuario necesita más ancho de banda o una mayor confiabilidad de la que le puede proporcionar una de las trayectorias de red, una solución es tener una conexión que distribuya el tráfico entre varias trayectorias de red por turno rotatorio (*round-robin*), como se indica en la figura 6-17(b). Este *modus operandi* se denomina **multiplexión inversa**. Con k conexiones de red abiertas, el ancho de banda efectivo se podría incrementar por un factor de k. El **SCTP** (**Protocolo de Control de Transmisión de Flujo**, del inglés *Stream Control Transmission Protocol*) es un ejemplo de multiplexión inversa. Este protocolo puede operar una conexión mediante el uso de múltiples interfaces de red. En contraste, TCP utiliza un solo punto terminal de red. La multiplexión inversa también se encuentra en la capa de enlace, cuando se usan varios enlaces de baja velocidad en paralelo como un solo enlace de alta velocidad.

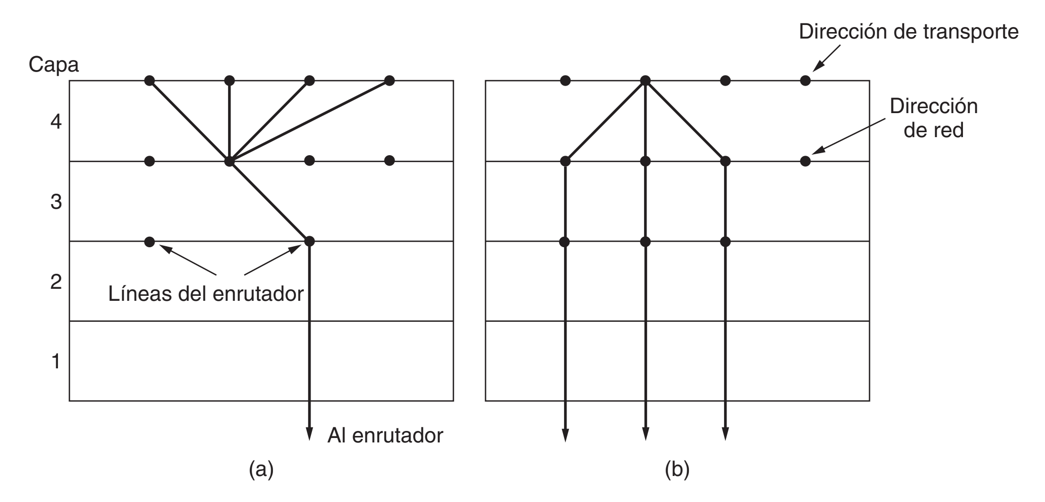


Figura 6-17. (a) Multiplexión. (b) Multiplexión inversa.

6.2.6 Recuperación de fallas

Si los hosts y los enrutadores están sujetos a fallas o las conexiones son de larga duración (por ejemplo, descargas extensas de software o medios), la recuperación de estas fallas se vuelve un tema importante. Si la entidad de transporte está totalmente dentro de los hosts, la recuperación de fallas de la red y de los enrutadores es sencilla. Las entidades de transporte esperan la pérdida de segmentos todo el tiempo y saben cómo lidiar con esto mediante el uso de retransmisiones.

Un problema más complicado es cómo recuperarse de las fallas del host. En particular, tal vez sea conveniente que los clientes sean capaces de continuar trabajando cuando los servidores fallan y se reinician muy rápido. Para ilustrar la dificultad, supongamos que un host, el cliente, envía un archivo grande a otro host, el servidor de archivos, mediante el uso de un protocolo simple de parada y espera. La capa de transporte en el servidor simplemente pasa los segmentos entrantes al usuario de transporte, uno por uno. De pronto, a mitad de la transmisión falla el servidor. Al reactivarse, sus tablas se reinicializan, por lo que ya no sabe precisamente en dónde se encontraba.

En un intento por recuperar su estado previo, el servidor podría enviar un segmento de difusión a todos los demás hosts, para anunciar que acaba de fallar y solicitar a sus clientes que le informen sobre el estado de todas las conexiones abiertas. Cada cliente puede estar en uno de dos estados: un segmento pendiente (*S1*) o ningún segmento pendiente (*S0*). Con base en esta información de estado, el cliente debe decidir si retransmitirá o no el segmento más reciente.

A primera vista parecería obvio: el cliente debe retransmitir sólo si tiene un segmento pendiente sin confirmación de recepción (es decir, que se encuentre en el estado *S1*) al momento de enterarse de la falla. Sin embargo, una inspección más cercana revela dificultades con esta metodología ingenua. Considere, por ejemplo, la situación en la que la entidad de transporte del servidor envía primero una confirmación de recepción y luego, una vez que se envía esta confirmación de recepción, escribe en el proceso de aplicación. Escribir un segmento en el flujo de salida y enviar una confirmación de recepción son dos eventos diferentes que no se pueden hacer al mismo tiempo. Si ocurre una falla después de enviar la confirmación de recepción pero antes de que la escritura se complete, el cliente recibirá la confirmación de recepción y estará por ende en el estado *S0* cuando llegue el anuncio de recuperación de la falla. Entonces el cliente no retransmitirá, pues pensará (en forma errónea) que llegó el segmento. Esta decisión del cliente provoca que falte un segmento.

En este punto el lector podría pensar: “Ese problema se resuelve fácil. Todo lo que hay que hacer es reprogramar la entidad de transporte para que primero haga la escritura y luego envíe la confirmación de recepción”. Intente de nuevo. Imagine que se ha hecho la escritura pero que la falla ocurre antes de enviar la confirmación de recepción. El cliente se encontrará en el estado *S1*, y por tanto retransmitirá, lo que provocará un segmento duplicado sin detectar en el flujo de salida que va al proceso de aplicación del servidor.

Sin importar cómo se programen el emisor y el receptor, siempre hay situaciones en las que el protocolo no se puede recuperar de manera apropiada. Podemos programar el servidor en una de dos formas: enviar confirmación de recepción primero o escribir primero. El cliente se puede programar en una de cuatro formas: siempre retransmitir el último segmento, nunca retransmitir el último segmento, retransmitir sólo en el estado *S0* o retransmitir sólo en el estado *S1*. Esto nos da ocho combinaciones pero, como veremos, para cada combinación existe cierto conjunto de eventos que hacen fallar al protocolo.

Son posibles tres eventos en el servidor: enviar una confirmación de recepción (*A*), escribir al proceso de salida (*W*) y fallar (C). Los tres eventos pueden ocurrir en seis órdenes diferentes: *AC*(*W*), *AWC*, *C*(*AW*), *C*(*WA*), *WAC* y *WC*(*A*), donde los paréntesis se usan para indicar que ni *A* ni *W* pueden ir después de *C* (es decir, una vez que el servidor falla, así se queda). En la figura 6-18 se muestran las ocho combinaciones de las estrategias de cliente y servidor, junto con las secuencias de eventos válidas para cada una. Observe que para cada estrategia hay alguna secuencia de eventos que provoca que el protocolo falle. Por ejemplo, si el cliente siempre retransmite, el evento *AWC* generará un duplicado no detectado, incluso aunque los otros dos eventos funcionen de manera apropiada.

	Estrategia que usa el host receptor					
	Primero ACK, luego escritura			Primero escritura, luego ACK		
Estrategia que usa el host emisor	AC(W)	AWC	C(AW)	C(WA)	W AC	WC(A)
Siempre retransmitir	BIEN	DUP	BIEN	BIEN	DUP	DUP
Nunca retransmitir	PERDIDO	BIEN	PERDIDO	PERDIDO	BIEN	BIEN
Retransmitir en S0	BIEN	DUP	PERDIDO	PERDIDO	DUP	BIEN
Retransmitir en S1	PERDIDO	BIEN	BIEN	BIEN	BIEN	DUP

BIEN = El protocolo funciona correctamente
DUP = El protocolo genera un mensaje duplicado
PERDIDO = El protocolo pierde un mensaje

Figura 6-18. Distintas combinaciones de estrategias de cliente y servidor.

Hacer más elaborado el protocolo no sirve de nada. Aunque el cliente y el servidor intercambien varios segmentos antes de que el servidor intente escribir, para que el cliente sepa con exactitud lo que está a punto de ocurrir, el cliente no tiene manera de saber si ha ocurrido una falla justo antes o justo después de la escritura. La conclusión es inevitable: según nuestra regla básica de que no debe haber eventos simultáneos (es decir, que eventos separados ocurren uno después de otro y no al mismo tiempo), la falla de un host y su recuperación no pueden hacerse transparentes a las capas superiores.

Dicho en términos más generales, podemos replantear este resultado como “la recuperación de una falla en la capa *N* sólo se puede llevar a cabo en la capa *N* + 1”, y esto es sólo si la capa superior retiene suficiente información del estado como para reconstruir la condición en la que se encontraba antes de que

ocurriera el problema. Esto es consistente con el caso que mencionamos antes, en el que la capa de transporte puede recuperarse de fallas en la capa de red, siempre y cuando cada extremo de una conexión lleve el registro del estado en el que se encuentra.

Este problema nos lleva a la cuestión de averiguar lo que en realidad significa una confirmación de recepción de extremo a extremo. En principio, el protocolo de transporte es de extremo a extremo y no está encadenado como en las capas inferiores. Ahora considere el caso de un usuario que introduce solicitudes de transacciones para una base de datos remota. Suponga que la entidad de transporte remota está programada para pasar primero los segmentos a la siguiente capa superior y luego emitir las confirmaciones de recepción. Incluso en este caso, el hecho de que la máquina de un usuario reciba una confirmación de recepción no significa necesariamente que el host remoto se quedó encendido el tiempo suficiente como para actualizar la base de datos. Quizá es imposible lograr una verdadera confirmación de recepción de extremo a extremo, en donde si se recibe significa que en realidad se hizo el trabajo y, si no se recibe, significa que no se realizó el trabajo. Este punto lo analizan con mayor detalle Saltzer y colaboradores (1984).

6.3 CONTROL DE CONGESTIÓN

Si las entidades de transporte en muchas máquinas envían demasiados paquetes a la red con exagerada rapidez, ésta se congestionará y se degradará el desempeño a medida que se retrasen y pierdan paquetes. El proceso de controlar la congestión para evitar este problema es la responsabilidad combinada de las capas de red y de transporte. La congestión ocurre en los enrutadores, por lo que se detecta en la capa de red. Sin embargo, se produce en última instancia debido al tráfico que la capa de transporte envía a la red. La única manera efectiva de controlar la congestión es que los protocolos de transporte envíen paquetes a la red con más lentitud.

En el capítulo 5 estudiamos los mecanismos de control de congestión en la capa de red. En esta sección estudiaremos la otra mitad del problema: los mecanismos de control de congestión en la capa de transporte. Después de describir los objetivos del control de congestión, describiremos la forma en que los hosts pueden regular la velocidad a la que envían los paquetes a la red. Internet depende mucho de la capa de transporte para el control de la congestión, por lo cual se han construido algoritmos específicos en TCP y otros protocolos.

6.3.1 Asignación de ancho de banda deseable

Antes de describir cómo regular el tráfico, debemos comprender lo que estamos tratando de lograr, al ejecutar un algoritmo de control de congestión. Esto es, debemos especificar el estado en el que un buen algoritmo de control de congestión operará la red. El objetivo es algo más que simplemente evitar la congestión. Se trata también de encontrar una buena asignación de ancho de banda a las entidades de transporte que utilizan la red. Una buena asignación producirá un buen desempeño, ya que utiliza todo el ancho de banda disponible pero evita la congestión, tratará con igualdad a las entidades de transporte que compitan entre sí, y rastreará con rapidez los cambios en las demandas de tráfico. A continuación explicaremos con más detalle cada uno de estos criterios por separado.

Eficiencia y potencia

Una asignación eficiente del ancho de banda entre las entidades de transporte utilizará toda la capacidad disponible de la red. Sin embargo, no es correcto pensar que si hay un enlace de 100 Mbps, cinco entidades

de transporte deben recibir 20 Mbps cada una. Por lo general deben recibir menos de 20 Mbps para un buen desempeño. La razón es que, con frecuencia, el tráfico es en ráfagas. Recuerde que en la sección 5.3 describimos el **caudal útil** (o tasa de paquetes útiles que llegan al receptor) como función de la carga ofrecida. En la figura 6-19 se muestran esta curva y otra similar para el retardo en función de la carga ofrecida.

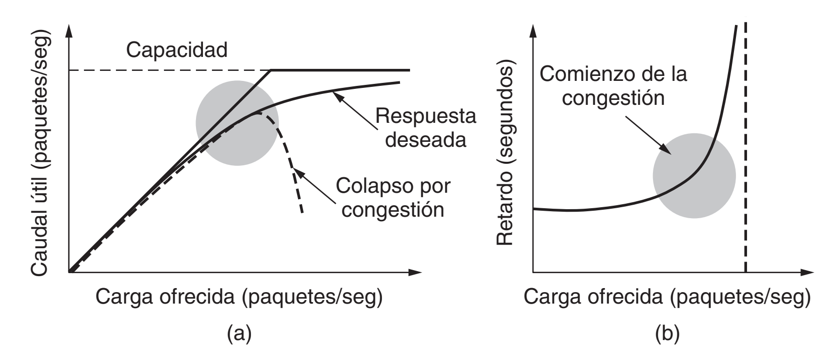


Figura 6-19. (a) Caudal útil y (b) retardo en función de la carga ofrecida.

A medida que aumenta la carga en la figura 6-19(a), el caudal útil aumenta en un principio con la misma proporción, pero a medida que la carga se acerca a la capacidad, el caudal útil aumenta en forma más gradual. Esta reducción se debe a que en ocasiones se pueden acumular las ráfagas de tráfico y provocar algunas pérdidas en los búferes dentro de la red. Si el protocolo de transporte está mal diseñado y retransmite paquetes que se hayan retardado pero que no se hayan perdido, la red puede entrar en un colapso por congestión. En este estado los emisores envían demasiados paquetes ya gran velocidad pero cada vez se realiza menos trabajo útil.

En la figura 6-19(b) se muestra el retardo correspondiente. En un principio, este retardo es fijo y representa el retardo de propagación a través de la red. A medida que la carga se acerca a la capacidad el retardo aumenta, lentamente al principio y después con mucha mayor rapidez. De nuevo, esto se debe a las ráfagas de tráfico que tienden a acumularse cuando la carga es alta. En realidad el retardo no puede llegar hasta infinito, excepto en un modelo en el que los enrutadores tengan búferes infinitos. En cambio, se perderán paquetes después de experimentar el retardo máximo en los búferes.

Tanto para el caudal útil como para el retardo, el desempeño se empieza a degradar cuando comienza la congestión. Por instinto, obtendremos el mejor desempeño de la red si asignamos ancho de banda hasta el punto en que el retardo empieza a aumentar con rapidez. Este punto está por debajo de la capacidad. Para identificarlo, Kleinrock (1979) propuso la métrica de **potencia**, en donde

$$Potencia = \frac{Carga}{Retardo}$$

En un principio la potencia aumentará con la carga ofrecida, mientras el retardo permanezca en un valor pequeño y aproximadamente constante, pero llegará a un máximo y caerá a medida que el retardo aumente con rapidez. La carga con la potencia más alta representa una carga eficiente para que la entidad de transporte la coloque en la red.

Equidad máxima-mínima

En la discusión anterior no hablamos sobre cómo dividir el ancho de banda entre distintos emisores de transporte. Esto suena como una pregunta simple de responder (dar a todos los emisores una misma fracción del ancho de banda), pero implica varias consideraciones.

Tal vez la primera consideración sea preguntar qué tiene que ver este problema con el control de la congestión. Después de todo, si la red proporciona a un emisor cierta cantidad de ancho de banda para que la utilice, el emisor debe usar sólo esa cantidad de ancho de banda. Sin embargo, es común el caso en que las redes no tienen una reservación estricta de ancho de banda para cada flujo o conexión. Tal vez sí la tengan para algunos flujos en los que se soporta la calidad del servicio, pero muchas conexiones buscarán usar el ancho de banda que esté disponible o la red las agrupará bajo una asignación común. Por ejemplo, los servicios diferenciados de la IETF separan el tráfico en dos clases y las conexiones compiten por el ancho de banda dentro de cada clase. A menudo, los enrutadores IP tienen conexiones que compiten por el mismo ancho de banda. En esta situación, el mecanismo de control de congestión es quien asigna el ancho de banda a las conexiones que compiten entre sí.

Una segunda consideración es lo que significa una porción equitativa para los flujos en una red. Es lo bastante simple si *N* flujos usan un solo enlace, en cuyo caso todos pueden tener 1/*N* parte del ancho de banda (aunque la eficiencia dictará que deben usar un poco menos, si el tráfico es en ráfagas). Pero ¿qué ocurre si los flujos tienen distintas trayectorias de red que se traslapan? Por ejemplo, un flujo puede atravesar tres enlaces y los otros flujos pueden atravesar un enlace. El flujo de tres enlaces consume más recursos de red. Puede ser más equitativo en cierto sentido para darle menos ancho de banda que a los flujos de un enlace. Sin duda debe ser posible soportar más flujos de un enlace al reducir el ancho de banda de tres enlaces. Este punto demuestra una tensión inherente entre la equidad y la eficiencia.

No obstante, adoptaremos una noción de equidad que no depende de la longitud de la trayectoria de red. Incluso con este simple modelo, es un poco complicado otorgar a las conexiones una fracción equitativa del ancho de banda, ya que las distintas conexiones tomarán distintas trayectorias a través de la red, y estas trayectorias tendrán distintas capacidades unas de otras. En este caso, es posible que un flujo sufra un congestionamiento en un enlace descendente y tome una porción más pequeña de un enlace ascendente que los otros flujos; reducir el ancho de banda de los otros flujos les permite disminuir su velocidad, pero no ayuda en nada al flujo con el congestionamiento.

La forma de equidad que se desea con frecuencia para el uso de red es la **equidad máxima-mínima**. Una asignación tiene equidad máxima-mínima si el ancho de banda que se otorga a un flujo no se puede incrementar sin tener que disminuir el ancho de banda otorgado a otro flujo con una asignación que no sea mayor. Esto es, si se incrementa el ancho de banda de un flujo, la situación sólo empeorará para los flujos con menos recursos.

Ahora veamos un ejemplo. En la figura 6-20 se muestra la asignación de equidad máxima-mínima para una red con cuatro flujos: *A*, *B*, *C* y *D*. Cada uno de los enlaces entre enrutadores tiene la misma capacidad, que se considera 1 unidad, aunque en el caso general los enlaces tendrán distintas capacidades. Tres flujos compiten por el enlace inferior izquierdo entre los enrutadores *R4* y *R5*. Por lo tanto, cada uno de estos flujos recibe 1/3 del enlace. El flujo restante *A* compite con *B* en el enlace de *R2* a *R3*. Como *B* tiene una asignación de 1/3, *A* recibe los 2/3 restantes del enlace. Observe que los demás enlaces tienen capacidad de sobra. Sin embargo, esta capacidad no se puede dar a ninguno de los flujos sin reducir la capacidad de otro flujo inferior. Por ejemplo, si se otorga más del ancho de banda en el enlace entre *R2* y *R3* al flujo *B*, habrá menos ancho de banda para el flujo *A*. Esto es razonable, puesto que el flujo *A* ya tiene de antemano más ancho de banda. Sin embargo, hay que reducir la capacidad del flujo *C* o *D* (o de ambos) para otorgar más ancho de banda a *B*, y estos flujos tendrán menor ancho de banda que *B*. Por ende, la asignación tiene equidad máxima-mínima.

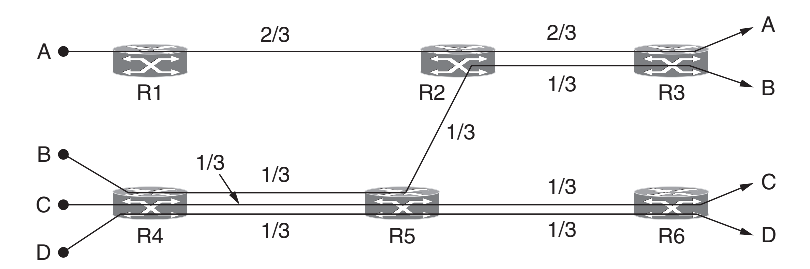


Figura 6-20. Asignación de ancho de banda con equidad máxima-mínima para cuatro flujos.

Las asignaciones máxima-mínima se pueden calcular con base en un conocimiento global de la red. Una forma intuitiva de pensar sobre ellas es imaginar que la tasa para todos los flujos empieza en cero y se incrementa lentamente. Cuando la tasa llega a un cuello de botella para cualquier flujo, entonces ese flujo deja de incrementarse. Los demás flujos seguirán incrementándose y compartirán la capacidad disponible por igual, hasta que también lleguen a sus respectivos cuellos de botella.

Una tercera consideración es el nivel sobre el cuál una red se puede considerar equitativa: al nivel de las conexiones, de las conexiones entre un par de hosts o de todas las conexiones por host. Ya examinamos esta cuestión cuando vimos el WFQ (Encolamiento justo ponderado) en la sección 5.4 y concluimos que cada una de estas definiciones tiene sus problemas. Por ejemplo, definir la equidad por host significa que un servidor ocupado se comportará igual que un teléfono móvil, mientras que al definir la equidad por conexión se anima a los hosts a que abran más conexiones. Dado que no hay una respuesta clara, lo más común es considerar la equidad por conexión, pero por lo general no es imprescindible una equidad precisa. Es más importante en la práctica que ninguna conexión se quede sin ancho de banda, a que todas las conexiones reciban la misma cantidad exacta de ancho de banda. De hecho, con TCP es posible abrir varias conexiones y competir por el ancho de banda en forma más agresiva. Esta táctica la utilizan aplicaciones que requieren grandes cantidades de ancho de banda, como BitTorrent para compartir archivos de igual a igual.

Convergencia

Un último criterio es que el algoritmo de control de congestión debe converger rápidamente hacia una asignación equitativa y eficiente del ancho de banda. El análisis anterior del punto de operación deseable supone un entorno de red estático. Sin embargo, en una red siempre entran y salen conexiones, por lo que el ancho de banda necesario para una conexión dada también variará con el tiempo; por ejemplo, cuando un usuario navega por páginas web y en algunas ocasiones descarga videos extensos.

Debido a la variación en la demanda, el punto de operación ideal para la red varía con el tiempo. Un buen algoritmo de control de congestión debe converger con rapidez hacia el punto de operación ideal; y debe rastrear ese punto a medida que cambia con el tiempo. Si la convergencia es muy lenta, el algoritmo nunca estará cerca del punto de operación cambiante. Si el algoritmo no es estable, puede fracasar al tratar de converger hacia el punto correcto en algunos casos, o incluso puede oscilar alrededor del punto correcto.

En la figura 6-21 se muestra un ejemplo de una asignación de ancho de banda que cambia con el tiempo y converge con rapidez. En un principio, el flujo 1 tiene todo el ancho de banda. Un segundo después, el flujo 2 empieza y también necesita ancho de banda. La asignación cambia rápidamente para otorgar a cada uno de estos flujos la mitad del ancho de banda. A los 4 segundos aparece un tercer flujo. Sin embargo, este flujo utiliza sólo el 20% del ancho de banda, lo cual es menos que su fracción equitativa (que es un tercio).

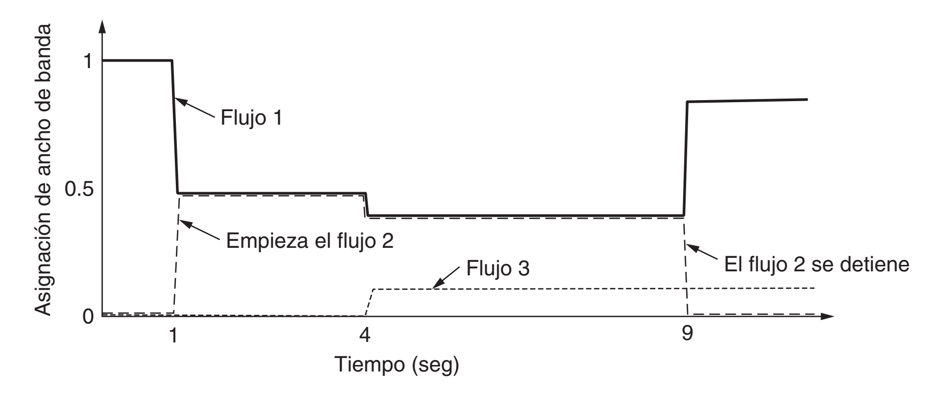


Figura 6-21. Ejemplo de una asignación de ancho de banda que cambia con el tiempo.

Los flujos 1 y 2 se ajustan rápidamente y dividen el ancho de banda disponible, para que cada uno tenga 40% del ancho de banda. A los 9 segundos termina el segundo flujo y el tercero permanece sin cambios. El primer flujo captura rápidamente 80% del ancho de banda. En todo momento, el ancho de banda total asignado es de aproximadamente 100%, de tal forma que la red se utilice por completo y los flujos que compiten obtengan un trato equitativo (pero no tienen que usar más ancho de banda del que necesitan).

6.3.2 Regulación de la tasa de envío

Ahora es tiempo para el platillo principal. ¿Cómo regulamos las tasas de envío para obtener una asignación de ancho de banda deseable? Podemos limitar la tasa de envío mediante dos factores. El primero es el control de flujo, en el caso en que haya un uso insuficiente de búfer en el receptor. El segundo es la congestión, en el caso en que haya una capacidad insuficiente en la red. En la figura 6-22 podemos ver este problema ilustrado mediante hidráulica. En la figura 6-22(a) vemos un tubo grueso que conduce a un receptor de baja capacidad. Ésta es una situación limitada por control de flujo. Mientras que el emisor no envíe más agua de la que pueda contener la cubeta, no se perderá agua. En la figura 6-22(b), el factor limitante no es la capacidad de la cubeta, sino la capacidad de transporte interno de la red. Si entra demasiada agua con mucha rapidez, se regresará y una parte se perderá (en este caso, por desbordamiento del embudo).

Estos casos pueden parecer similares para el emisor, puesto que al transmitir demasiado rápido se pierden paquetes. Sin embargo, tienen distintas causas y requieren diferentes soluciones. Ya hemos hablado sobre una solución de control de flujo con una ventana de tamaño variable. Ahora consideraremos una solución de control de congestión. Como puede ocurrir cualquiera de estos problemas, en general el protocolo de transporte tendrá que llevar a cabo ambas soluciones y reducir la velocidad si ocurre cualquiera de los dos problemas.

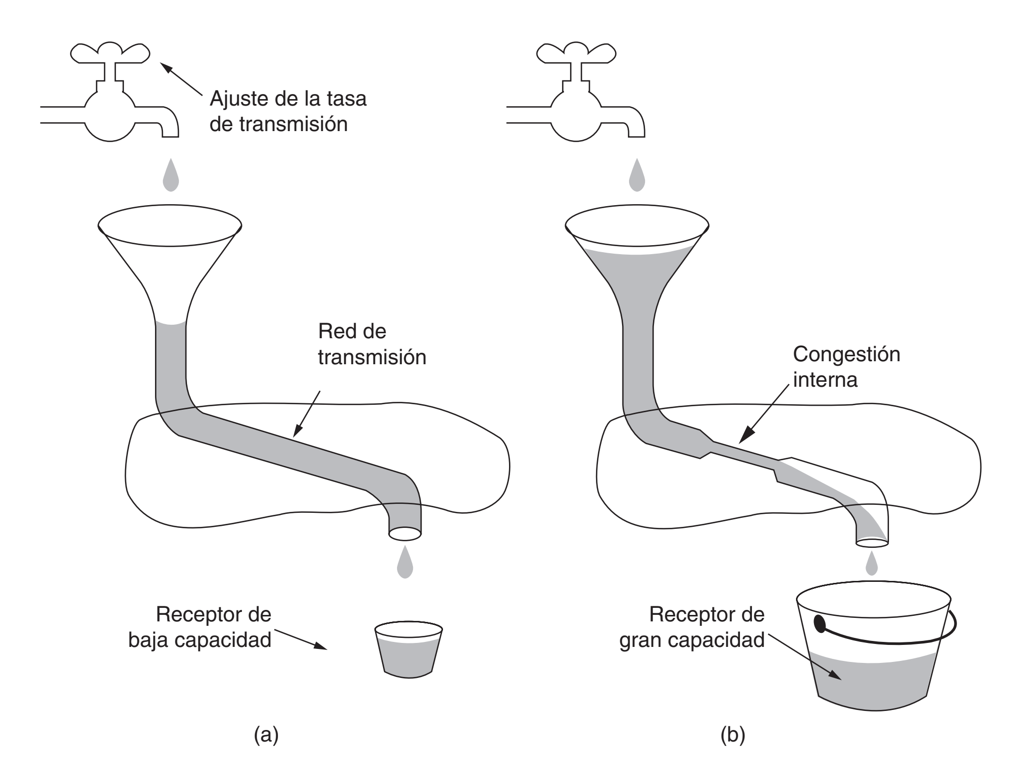


Figura 6-22. (a) Una red veloz que alimenta a un receptor de baja capacidad. (b) Una red lenta que alimenta a un receptor de alta capacidad.

La forma en que un protocolo de transporte debe regular la tasa de envío depende de la forma de retroalimentación que devuelva la red. Las distintas capas de la red pueden devolver distintos tipos de retroalimentación, la cual puede ser explícita o implícita, y también precisa o imprecisa.

Un ejemplo de un diseño explícito y preciso es cuando los enrutadores indican a las fuentes la velocidad a la que pueden enviar. Los diseños en la literatura, como XCP (Protocolo Explícito de Congestión, del inglés *eXplicit Congestion Protocol*), operan de esta forma (Katabi y colaboradores, 2002). Un diseño explícito e impreciso es el uso de ECN (Notificación Explícita de Congestión, del inglés *Explicit Congestion Notification*) con TCP. En este diseño, los enrutadores establecen bits en paquetes que experimentan congestión para advertir a los emisores que reduzcan la velocidad, pero no les dicen cuánto deben reducirla.

En otros diseños no hay una señal explícita. FAST TCP mide el retardo de ida y vuelta, y usa esa métrica como señal para evitar la congestión (Wei y colaboradores, 2006). Por último, en la forma de control de congestión más prevalente en Internet en la actualidad, TCP con enrutadores *drop-tail* (descartar final) o RED, se deduce la pérdida de paquetes y se utiliza para indicar que esa red se ha congestionado. Existen muchas variantes de esta forma de TCP, incluyendo CUBIC TCP, que se utiliza en Linux (Ha y colaboradores, 2008). También son posibles las combinaciones. Por ejemplo, Windows incluye el diseño Compound TCP, que utiliza la pérdida de paquetes y el retardo como señales de retroalimentación (Tan y colaboradores, 2006). En la figura 6-23 se resumen estos diseños.

Si se proporciona una señal explícita y precisa, la entidad de transporte puede usarla para ajustar su tasa con el nuevo punto de operación. Por ejemplo, si XCP indica a los emisores la tasa que deben usar, éstos simplemente usarán esa tasa. Sin embargo, en los otros casos se requieren algunas conjeturas. A falta de una señal de congestión, los emisores deben reducir sus tasas. Cuando se proporcione una señal de congestión, los emisores deben reducir sus tasas. La forma en que se deben aumentar o reducir las tasas se proporciona mediante una **ley de control**. Estas leyes tienen un efecto importante en el desempeño.

Protocolo	Señal	¿Explícito?	¿Preciso?
XCP	Tasa a usar.	Sí	Sí
TCP con ECN	Advertencia de congestión.	Sí	No
FAST TCP	Retardo de extremo a extremo.	No	Sí
Compound TCP	Pérdida de paquete y retardo extremo a extremo.	No	Sí
CUBIC TCP	Pérdida de paquetes.	No	No
TCP	Pérdida de paquetes.	No	No

Figura 6-23. Señales de algunos protocolos de control de congestión.

Chiu y Jain (1989) estudiaron el caso de la retroalimentación por congestión binaria y concluyeron que **AIMD** (**Incremento Aditivo/Decremento Multiplicativo**, del inglés *Additive Increase Multiplicative Decrease*) es la ley de control apropiada para llegar a un punto de operación eficiente y equitativo.

Para exponer este caso, construyeron un argumento gráfico para el caso simple de dos conexiones que compiten por el ancho de banda de un solo enlace. El gráfico de la figura 6-24 muestra el ancho de banda asignado al usuario 1 en el eje x y al usuario 2 en el eje y. Cuando la asignación es equitativa, ambos usuarios reciben la misma cantidad de ancho de banda. Esto se muestra mediante la línea de equidad punteada. Cuando las asignaciones suman un 100% de la capacidad del enlace, la asignación es eficiente. Esto se muestra mediante la línea de eficiencia punteada. La red proporciona una señal de congestión a ambos usuarios cuando la suma de sus asignaciones cruza esta línea. La intersección de esas líneas es el punto de operación deseado, cuando ambos usuarios tienen el mismo ancho de banda y se utiliza todo el ancho de banda de la red.

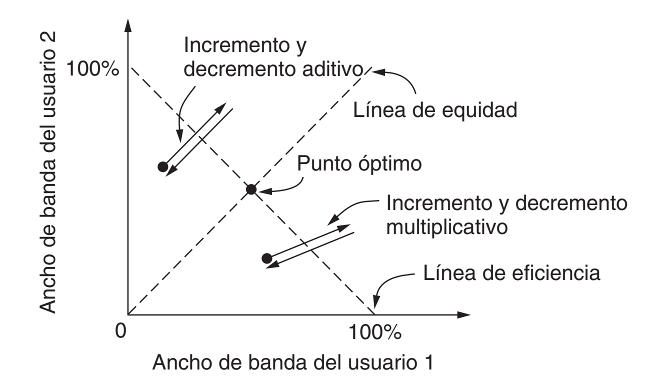


Figura 6-24. Ajustes de ancho de banda aditivo y multiplicativo.

Considere lo que ocurre desde alguna asignación inicial si tanto el usuario 1 como el 2 incrementan en forma aditiva su respectivo ancho de banda a través del tiempo. Por ejemplo, cada uno de los usuarios puede incrementar su tasa de envío por 1 Mbps cada segundo. En un momento dado, el punto de operación cruza la línea de eficiencia y ambos usuarios reciben de la red una señal de congestión. En esta etapa deben reducir sus asignaciones. Sin embargo, un decremento aditivo simplemente provocaría que oscilaran a lo largo de una línea aditiva. Esta situación se muestra en la figura 6-24. El comportamiento mantendrá el punto de operación cerca de lo eficiente, pero no necesariamente será equitativo.

De manera similar, considere el caso en que ambos usuarios incrementan en forma multiplicativa su ancho de banda a través del tiempo, hasta que reciben una señal de congestión. Por ejemplo, los usuarios pueden incrementar su tasa de envío en 10% cada segundo. Si después decrementan en forma multiplicativa sus tasas de envío, el punto de operación de los usuarios simplemente oscilará a lo largo de una línea multiplicativa. Este comportamiento también se muestra en la figura 6-24. La línea multiplicativa tiene una pendiente diferente a la línea aditiva (apunta hacia el origen, mientras que la línea aditiva tiene un ángulo de 45 grados). Pero de ninguna otra forma es mejor. En ningún caso los usuarios convergerán hacia las tasas de envío óptimas que sean tanto equitativas como eficientes.

Ahora considere el caso en que los usuarios incrementan en forma aditiva sus asignaciones de ancho de banda y después las decrementan en forma multiplicativa cuando se indica una congestión. Este comportamiento es la ley de control AIMD y se muestra en la figura 6-25. Podemos ver que la trayectoria trazada por este comportamiento converge hacia el punto óptimo, que es tanto equitativo como eficiente. Esta convergencia ocurre sin importar cuál sea el punto inicial, por lo cual el AIMD se considera en extremo útil. Con base en el mismo argumento, la única combinación restante, incremento multiplicativo y decremento aditivo, divergirá del punto óptimo.

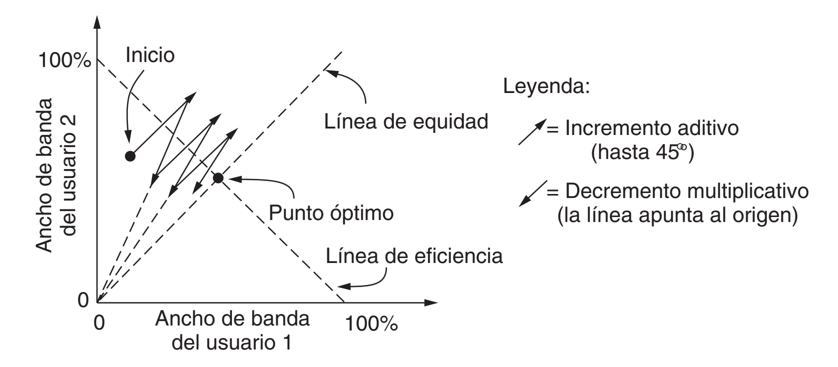


Figura 6-25. La ley de control de Incremento Aditivo/Decremento Multiplicativo (AIMD).

AIMD es la ley de control que utiliza TCP, con base en este argumento y en otro razonamiento de estabilidad (que es fácil llevar a la red a la congestión y difícil recuperarse, por lo que la política de incremento debe ser gentil y la política de decremento agresiva). No es muy equitativa, ya que las conexiones TCP ajustan el tamaño de su ventana por cierta cantidad en cada tiempo de ida y vuelta. Las distintas conexiones tendrán distintos tiempos de ida y vuelta. Esto conduce a una predisposición en donde las conexiones a los hosts cercanos reciben más ancho de banda que las conexiones a los hosts distantes, siendo todo lo demás igual.

En la sección 6.5 describiremos con detalle la forma en que TCP implementa una ley de control AIMD para ajustar la tasa de envío y proveer control de congestión. Esta tarea es más difícil de lo que parece, ya que las tasas se miden a través de cierto intervalo y el tráfico es en ráfagas. En vez de ajustar la tasa directamente, una estrategia de uso común en la práctica es ajustar el tamaño de una ventana deslizante. TCP usa esta estrategia. Si el tamaño de la ventana es *W* y el tiempo de ida y vuelta es *RTT*, la tasa equivalente es *W*/*RTT*. Esta estrategia es fácil de combinar con el control de flujo, que de antemano usa una ventana, además de que tiene la ventaja de que el emisor controla los paquetes usando confirmaciones de recepción y, por ende, reduce la velocidad en un *RTT* si deja de recibir los informes de que los paquetes están saliendo de la red.

Como cuestión final, puede haber muchos protocolos de transporte distintos que envíen tráfico a la red. ¿Qué ocurrirá si los distintos protocolos compiten con diferentes leyes de control para evitar la congestión? Se producirán asignaciones de ancho de banda desiguales. Como TCP es la forma dominante de control de congestión en Internet, hay una presión considerable de la comunidad en cuanto a diseñar nuevos protocolos de transporte para que compitan de manera equitativa con TCP. Los primeros protocolos de medios de flujo continuo provocaron problemas al reducir de manera excesiva la tasa de transmisión de TCP debido a que no competían en forma equitativa. Esto condujo a la noción del control de congestión **TCP-friendly** (amigable para TCP) , en donde se pueden mezclar los protocolos de transporte que sean o no de TCP sin efectos dañinos (Floyd y colaboradores, 2000).

6.3.3 Cuestiones inalámbricas

Los protocolos de transporte como TCP que implementan control de congestión deben ser independientes de la red subyacente y de las tecnologías de capa de enlace. Es una buena teoría, pero en la práctica hay problemas con las redes inalámbricas. La cuestión principal es que la pérdida de paquetes se usa con frecuencia como señal de congestión, incluyendo a TCP como vimos antes. Las redes inalámbricas pierden paquetes todo el tiempo debido a errores de transmisión.

Con la ley de control AIMD, una tasa de transmisión real alta requiere niveles muy pequeños de pérdida de paquetes. Los análisis realizados por Padhye y colaboradores (1998) mostraron que la tasa

de transmisión real aumenta con base en la raíz cuadrada inversa de la tasa de pérdida de paquetes. Lo que esto significa en la práctica es que la tasa de pérdidas para las conexiones TCP rápidas es muy pequeña: 1% es una tasa de pérdidas moderada, y para cuando la tasa de pérdidas llega a 10%, la conexión ha dejado de trabajar por completo. Sin embargo, para las redes inalámbricas como las LAN 802.11, son comunes las tasas de pérdidas de tramas de por lo menos un 10%. Esta diferencia significa que, sin medidas de protección, los esquemas de control de congestión que utilizan la pérdida de paquetes como señal regularán de manera innecesaria las conexiones que pasen a través de enlaces inalámbricos para generar tasas muy bajas.

Para funcionar bien, las únicas pérdidas de paquetes que debe observar el algoritmo de control de congestión son las pérdidas debido a un ancho de banda insuficiente, no las pérdidas debido a errores de transmisión. Una solución a este problema es enmascarar las pérdidas inalámbricas mediante el uso de retransmisiones a través del enlace inalámbrico. Por ejemplo, 802.11 usa un protocolo de parada y espera para entregar cada trama, y reintenta las transmisiones varias veces si es necesario antes de reportar la pérdida de un paquete a la capa superior. En el caso normal, cada paquete se entrega a pesar de los errores de transmisión transitorios que no son visibles para las capas superiores.

La figura 6-26 muestra una trayectoria con un enlace cableado y uno inalámbrico, para el que se utiliza la estrategia de enmascaramiento. Hay que tener en cuenta dos aspectos. En primer lugar, el emisor no necesariamente sabe que la trayectoria incluye un enlace inalámbrico, ya que todo lo que ve es el enlace cableado al que está conectado. Las trayectorias de Internet son heterogéneas, por lo que no hay un método general para que el emisor sepa qué tipo de enlaces conforman la trayectoria. Esto complica el problema de control de congestión, ya que no hay una manera fácil de usar un protocolo para enlaces inalámbricos y otro para enlaces cableados.

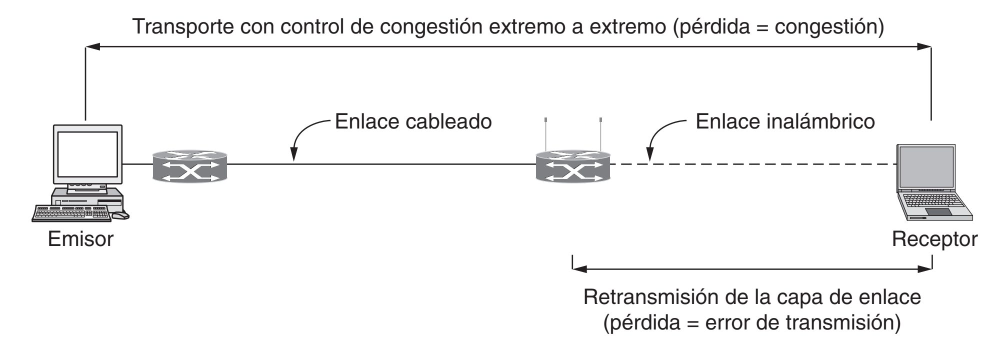


Figura 6-26. Control de congestión a través de una trayectoria con un enlace inalámbrico.

El segundo aspecto es un acertijo. La figura muestra dos mecanismos controlados por la pérdida: las retransmisiones de tramas de la capa de enlace y el control de congestión de la capa de transporte. El acertijo es: cómo pueden coexistir estos dos mecanismos sin confundirse. Después de todo, una pérdida sólo debe provocar que un mecanismo actúe, ya que puede ser un error de transmisión o una señal de congestión. No puede ser ambos. Si los dos mecanismos actúan (al retransmitir la trama y reducir la tasa de envío), entonces regresamos al problema original de los transportes que operan con demasiada lentitud a través de los enlaces inalámbricos. Considere este acertijo por un momento y vea si puede resolverlo.

La solución es que los dos mecanismos actúan en distintas escalas de tiempo. Las retransmisiones de la capa de enlace ocurren en el orden de microsegundos a milisegundos para los enlaces inalámbricos como 802.11. Los temporizadores de pérdidas en los protocolos de transporte se activan en el orden de milisegundos a segundos. La diferencia es de tres órdenes de magnitud. Esto permite a los enlaces ina-

lámbricos detectar las pérdidas de tramas y retransmitirlas para reparar los errores de transmisión mucho antes de que la entidad de transporte deduzca la pérdida de paquetes.

La estrategia de enmascaramiento es suficiente para permitir que la mayoría de los protocolos de transporte operen bien a través de la mayoría de los enlaces inalámbricos. Sin embargo, no siempre es una solución adecuada. Algunos enlaces inalámbricos tienen tiempos de ida y vuelta largos, como los satélites. Para estos enlaces se deben usar otras técnicas para enmascarar la pérdida, como FEC (Corrección de Errores hacia Adelante), o el protocolo de transporte debe usar una señal que no sea de pérdida para el control de congestión.

Un segundo aspecto relacionado con el control de congestión a través de enlaces inalámbricos es la capacidad variable. Esto es, la capacidad de un enlace inalámbrico cambia con el tiempo, algunas veces en forma brusca, a medida que los nodos se desplazan y la relación señal —ruido varía con las condiciones cambiantes del canal. Esto es distinto a los enlaces cableados, cuya capacidad es fija. El protocolo de transporte se debe adaptar a la capacidad cambiante de los enlaces inalámbricos; de lo contrario se congestionará la red o no se podrá usar la capacidad disponible.

Una posible solución a este problema es simplemente no preocuparse por ello. Esta estrategia es viable, puesto que los algoritmos de control de congestión ya deben manejar el caso en que nuevos usuarios entren a la red, o que los usuarios existentes cambien sus tasas de envío. Aun cuando la capacidad de los enlaces cableados es fija, el comportamiento cambiante de otros usuarios se presenta a sí mismo como una variabilidad en el ancho de banda que está disponible para un usuario dado. Por ende, es posible usar TCP a través de una ruta con un enlace inalámbrico 802.11 y obtener un desempeño razonable.

Sin embargo, cuando hay mucha variabilidad inalámbrica, los protocolos de transporte diseñados para los enlaces cableados pueden tener problemas para mantenerse a la par y provocar un desempeño deficiente. La solución en este caso es un protocolo de transporte que esté diseñado para enlaces inalámbricos. Una configuración muy desafiante es una red de malla inalámbrica en la que se deben atravesar varios enlaces inalámbricos, es necesario cambiar rutas debido a la movilidad y muchas pérdidas. La investigación en esta área es continua. Consulte a Li y colaboradores (2009) para obtener un ejemplo de diseño de un protocolo de transporte inalámbrico.

6.4 LOS PROTOCOLOS DE TRANSPORTE DE INTERNET: UDP

Internet tiene dos protocolos principales en la capa de transporte, uno sin conexión y otro orientado a conexión. Los protocolos se complementan entre sí. El protocolo sin conexión es UDP. Prácticamente no hace nada más que enviar paquetes entre aplicaciones, y deja que las aplicaciones construyan sus propios protocolos en la parte superior según sea necesario. El protocolo orientado a conexión es TCP. Hace casi todo. Realiza las conexiones y agrega confiabilidad mediante las retransmisiones, junto con el control de flujo y el control de congestión, todo en beneficio de las aplicaciones que lo utilizan.

En las siguientes secciones estudiaremos los protocolos UDP y TCP. Empezaremos con UDP porque es el más simple. También veremos dos aplicaciones de UDP. Como éste es un protocolo de capa de transporte que por lo general se ejecuta en el sistema operativo y los protocolos que utilizan UDP por lo general se ejecutan en el espacio de usuario, estos usos se podrían considerar como aplicaciones. Sin embargo, las técnicas que utilizan son convenientes para muchas aplicaciones y se considera que pertenecen a un servicio de transporte, por lo que las veremos aquí.

6.4.1 Introducción a UDP

La suite de protocolos de Internet soporta un protocolo de transporte sin conexión, conocido como **UDP** (**Protocolo de Datagrama de Usuario**, del inglés *Use Datagram Protocol*). UDP proporciona una forma

para que las aplicaciones envíen datagramas IP encapsulados sin tener que establecer una conexión. El protocolo UDP se describe en el RFC 768.

UDP transmite **segmentos** que consisten en un encabezado de 8 bytes seguido de la carga útil. En la figura 6-27 se muestra ese encabezado. Los dos **puertos** sirven para identificar los puntos terminales dentro de las máquinas de origen y destino. Cuando llega un paquete UDP, su carga útil se entrega al proceso que está conectado al puerto de destino. Este enlace ocurre cuando se utiliza la primitiva BIND o algo similar, como vimos en la figura 6-6 para TCP (el proceso de enlace es el mismo para UDP). Piense en los puertos como apartados postales que las aplicaciones pueden rentar para recibir paquetes. Diremos más sobre ellas cuando describamos a TCP, que también usa puertos. De hecho, el valor principal de contar con UDP en lugar de simplemente utilizar IP puro es la adición de los puertos de origen y destino. Sin los campos de puerto, la capa de transporte no sabría qué hacer con cada paquete entrante. Con ellos, entrega el segmento incrustado a la aplicación correcta.

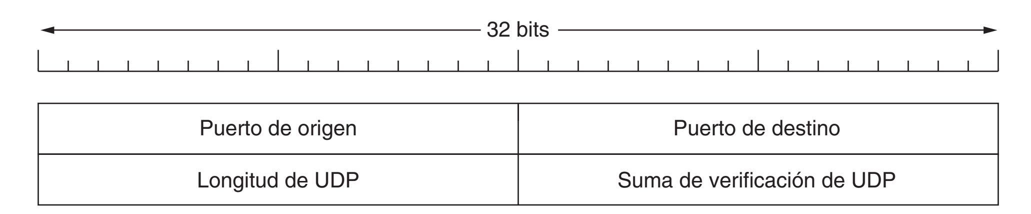


Figura 6-27. El encabezado UDP.

El puerto de origen se necesita principalmente cuando hay que enviar una respuesta al origen. Al copiar el campo *Puerto de origen* del segmento entrante en el campo *Puerto de destino* del segmento que sale, el proceso que envía la respuesta puede especificar cuál proceso de la máquina emisora va a recibirlo.

El campo *Longitud UDP* incluye el encabezado de 8 bytes y los datos. La longitud mínima es de 8 bytes, para cubrir el encabezado. La longitud máxima es de 65 515 bytes, lo cual es menor al número que cabe en 16 bits debido al límite de tamaño en los paquetes IP.

También se proporciona un campo *Suma de verificación* opcional para una confiabilidad adicional. Se realiza una suma de verificación para el encabezado, los datos y un pseudoencabezado IP conceptual. Al hacer este cálculo, el campo *Suma de verificación* se establece en cero y el campo de datos se rellena con un byte cero adicional si su longitud es un número impar. El algoritmo de suma de verificación consiste simplemente en sumar todas las palabras de 16 bits en complemento a uno y sacar el complemento a uno de la suma. Como consecuencia, cuando el receptor realiza el cálculo en todo el segmento (incluyendo el campo *Suma de verificación*), el resultado debe ser 0. Si no se calcula la suma de verificación, se almacena como 0, ya que por una feliz coincidencia de la aritmética de complemento a uno, un 0 calculado se almacena como 1. Sin embargo, no tiene sentido desactivarlo a menos que no importe la calidad de los datos (por ejemplo, en la voz digitalizada).

En la figura 6-28 se muestra el pseudoencabezado para el caso de IPv4. Éste contiene las direcciones IPv4 de 32 bits de las máquinas de origen y de destino, el número de protocolo para UDP (17) y la cuenta de bytes para el segmento UDP (incluyendo el encabezado). Es distinto pero análogo a IPv6. Es útil incluir el pseudoencabezado en el cálculo de la suma de verificación de UDP para detectar paquetes mal entregados, pero al incluirlo también se viola la jerarquía de protocolos debido a que las direcciones IP en él pertenecen a la capa IP, no a la capa UDP. TCP usa el mismo pseudoencabezado para su suma de verificación.

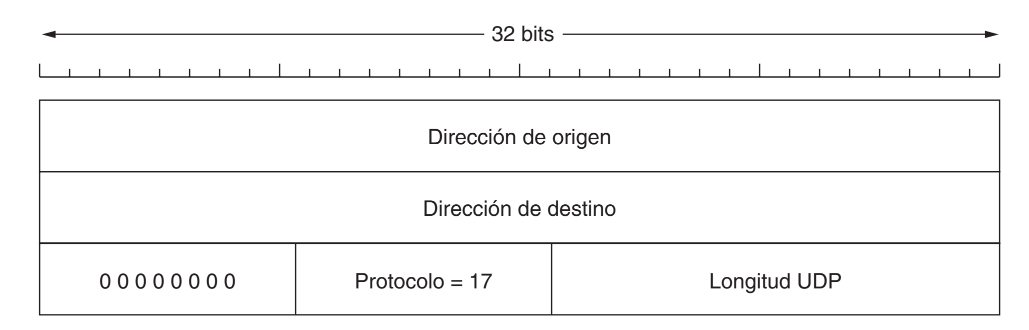


Figura 6-28. El pseudoencabezado IPv4 que se incluye en la suma de verificación de UDP.

Quizá valga la pena mencionar de manera explícita algunas de las cosas que UDP *no* realiza. No realiza control de flujo, control de congestión o retransmisión cuando se recibe un segmento erróneo. Todo lo anterior le corresponde a los procesos de usuario. Lo que sí realiza es proporcionar una interfaz para el protocolo IP con la característica agregada de demultiplexar varios procesos mediante el uso de los puertos y la detección de errores extremo a extremo opcional. Esto es todo lo que hace.

Para aplicaciones que necesitan tener un control preciso sobre el flujo de paquetes, control de errores o temporización, UDP es justo lo que se necesita. Un área en la que UDP es especialmente útil es en las situaciones cliente-servidor. Con frecuencia, el cliente envía una solicitud corta al servidor y espera una respuesta corta. Si se pierde la solicitud o la respuesta, el cliente simplemente puede esperar a que expire su temporizador e intentar de nuevo. El código no sólo es simple, sino que se requieren menos mensajes (uno en cada dirección) en comparación con un protocolo que requiere una configuración inicial, como TCP.

Una aplicación que utiliza de esta manera a UDP es DNS (el Sistema de Nombres de Dominio), el cual analizaremos en el capítulo 7. En resumen, un programa que necesita buscar la dirección IP de algún host, por ejemplo, www.cs.berkeley.edu, puede enviar al servidor DNS un paquete UDP que contenga el nombre de dicho host. El servidor responde con un paquete UDP que contiene la dirección IP del host. No se necesita configuración por adelantado ni tampoco una liberación posterior. Sólo dos mensajes que viajan a través de la red.

6.4.2 Llamada a procedimiento remoto

En cierto sentido, enviar un mensaje a un host remoto y obtener una respuesta es muy parecido a realizar la llamada a una función en un lenguaje de programación. En ambos casos, usted inicia con uno o más parámetros y obtiene un resultado. Esta observación ha llevado a la gente a tratar de que las interacciones de solicitud-respuesta en las redes se asignen en forma de llamadas a procedimientos. Dicho arreglo hace que las aplicaciones de red sean mucho más fáciles de programar y de manejar. Por ejemplo, imagine un procedimiento llamado *obtener_direccion_IP (nombre_de_host)* que envía un paquete UDP a un servidor DNS y espera una respuesta, y en caso de que no llegue ninguna con la suficiente rapidez, expira su temporizador y lo intenta de nuevo. De esta forma, todos los detalles de la conectividad pueden ocultarse al programador.

El trabajo clave en esta área fue realizado por Birrell y Nelson (1984). En sí, lo que ellos sugirieron fue permitir que los programas invocaran procedimientos localizados en hosts remotos. Cuando un proceso en la máquina 1 llama a otro procedimiento en la máquina 2, el proceso invocador en 1 se suspende y la ejecución del procedimiento invocado se lleva a cabo en la máquina 2. Se puede transportar información del proceso invocador al proceso invocado en los parámetros, y se puede regresar información en el

resultado del procedimiento. El paso de mensajes es transparente para el programador de la aplicación. Esta técnica se conoce como **RPC** (**Llamada a Procedimiento Remoto**, del inglés *Remote Procedure Call*) y se ha vuelto la base de muchas aplicaciones de red. Por tradición, el procedimiento invocador se conoce como cliente y el proceso invocado como servidor, por lo que aquí también utilizaremos esos nombres.

El objetivo de RPC es hacer que una llamada a procedimiento remoto sea lo más parecida posible a una local. En la forma más simple, para llamar a un procedimiento remoto, el programa cliente se debe enlazar con un pequeño procedimiento de biblioteca, llamado **stub del cliente**, que representa al procedimiento servidor en el espacio de direcciones del cliente. Asimismo, el servidor se enlaza con una llamada a procedimiento denominada **stub del servidor**. Estos procedimientos ocultan el hecho de que la llamada a procedimiento del cliente al servidor no es local.

En la figura 6-29 se muestran los pasos reales para realizar una RPC. El paso 1 consiste en que el cliente llame al stub del cliente. Ésta es una llamada a procedimiento local, y los parámetros se meten en la pila de la forma tradicional. El paso 2 consiste en que el stub del cliente empaque los parámetros en un mensaje y realice una llamada de sistema para enviar el mensaje. Al proceso de empacar los parámetros se le conoce como **marshaling (empaquetar)**. El paso 3 consiste en que el sistema operativo envíe el mensaje de la máquina cliente a la máquina servidor. El paso 4 consiste en que el sistema operativo pase el paquete entrante al stub del servidor. Por último, el paso 5 consiste en que el stub del servidor llame al procedimiento servidor con los parámetros sin empacar (*unmarshaling*). La respuesta sigue la misma ruta en la dirección opuesta.

El elemento clave a observar aquí es que el procedimiento cliente, escrito por el usuario, simplemente realiza una llamada a procedimiento normal (es decir, local) al stub del cliente, que tiene el mismo nombre que el procedimiento servidor. Puesto que el procedimiento cliente y el stub del cliente están en el mismo espacio de direcciones, los parámetros se pasan de la forma usual. De manera similar, el procedimiento servidor es llamado por un procedimiento en su espacio de direcciones con los parámetros esperados. Para el procedimiento servidor, nada es inusual. De esta forma, en lugar de que la E/S se realice en sockets, la comunicación de red se realiza mediante la simulación de una llamada a procedimiento normal.

A pesar de la elegancia conceptual de RPC, hay algunas desventajas ocultas. La más grande es el uso de parámetros de apuntador. Por lo general, no hay problema al pasar un apuntador a un procedimiento. El procedimiento invocado puede utilizar el apuntador de la misma manera que el invocador, porque ambos procedimientos se encuentran en el mismo espacio de direcciones virtual. Con RPC, el paso de apuntadores es imposible porque el cliente y el servidor están en diferentes espacios de direcciones.

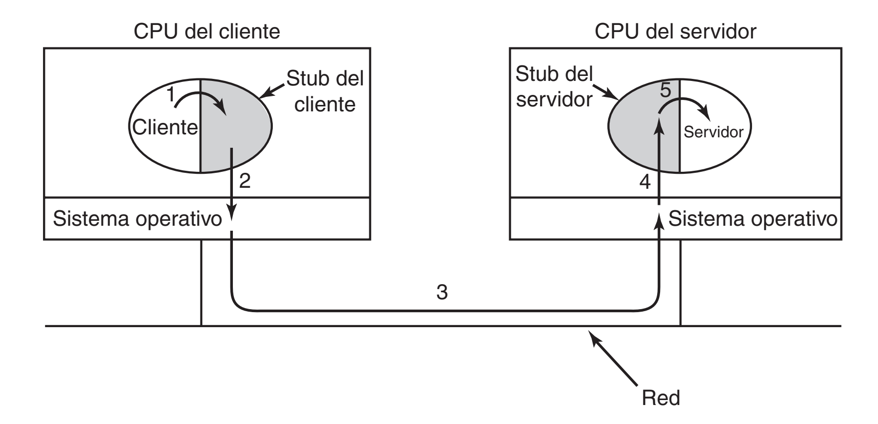


Figura 6-29. Pasos para realizar la llamada a un procedimiento remoto. Los stubs están sombreados.

En algunos casos, se pueden utilizar trucos para pasar apuntadores. Suponga que el primer parámetro es un apuntador a un entero, *k*. El stub del cliente puede empacar a *k* y enviarlo a lo largo del servidor. A continuación, el stub del servidor crea un apuntador a *k* y lo pasa al procedimiento servidor, justo como lo espera. Cuando el procedimiento servidor regresa el control al stub del servidor, este último envía a *k* de regreso al cliente, en donde el nuevo *k* se copia sobre el anterior, por si el servidor lo cambió. En efecto, la secuencia estándar de llamada por referencia se ha reemplazado por una llamada mediante copiar-restaurar. Por desgracia, este truco no siempre funciona; por ejemplo, si el apuntador apunta hacia un grafo u otra estructura de datos compleja. Por esta razón, se deben colocar algunas restricciones sobre los parámetros para los procedimientos que se llaman de manera remota, como veremos más adelante.

Un segundo problema es que en los lenguajes de tipos flexibles, como C, es perfectamente legal escribir un procedimiento que calcule el producto interno de dos vectores (arreglos), sin especificar la longitud de cada uno. Estos vectores se pueden terminar mediante un valor especial conocido sólo por los procedimientos invocador e invocado. Bajo estas circunstancias, es en esencia imposible que el stub del cliente empaquete los parámetros debido a que no tiene forma de determinar su longitud.

Un tercer problema es que no siempre es posible deducir los tipos de los parámetros, ni siquiera mediante una especificación formal o del mismo código. Un ejemplo de esto es el procedimiento *printf*, el cual puede tener cualquier número de parámetros (por lo menos uno), y éstos pueden ser una mezcla arbitraria de tipos enteros, cortos, largos, caracteres, cadenas, así como de números de punto flotante de varias longitudes, entre otros. Tratar de llamar a *printf* como un procedimiento remoto sería prácticamente imposible debido a que C es demasiado permisivo. Sin embargo, una regla que especifique el uso de RPC, siempre y cuando no se utilice C (o C++) para programar, tal vez no sea muy popular con muchos programadores.

Un cuarto problema se relaciona con el uso de las variables globales. Por lo general, los procedimientos invocador e invocado se pueden comunicar mediante el uso de variables globales, además de comunicarse a través de los parámetros. Pero si el procedimiento invocado se mueve a una máquina remota, el código fallará porque las variables globales ya no estarán compartidas.

Estos problemas no quieren decir que RPC no tenga mucho futuro. De hecho, se utiliza ampliamente, pero se necesitan algunas restricciones para hacerlo funcionar bien en la práctica.

En términos de protocolos de capa de transporte, UDP es una buena base sobre la cual se puede implementar la técnica RPC. Se pueden enviar tanto solicitudes como respuestas en un solo paquete UDP en el caso más simple; además la operación puede ser rápida. Sin embargo, una implementación debe incluir también otras herramientas. Como se puede llegar a perder la solicitud o la respuesta, el cliente debe mantener un temporizador para retransmitir la solicitud. Debemos tener en cuenta que una respuesta sirve como una confirmación de recepción explícita de una solicitud, por lo que no es necesario confirmar la recepción de la solicitud por separado. Algunas veces los parámetros o resultados pueden ser más grandes que el tamaño de paquete UDP máximo, en cuyo caso se requiere de algún protocolo para entregar mensajes grandes. Si existe la probabilidad de que varias solicitudes y respuestas se traslapen (como en el caso de la programación concurrente), se necesita un identificador para relacionar la solicitud con la respuesta.

Una preocupación de mayor nivel es que la operación tal vez no sea idempotente (es decir, que se pueda repetir en forma segura). El caso simple es el de las operaciones idempotentes como las solicitudes y respuestas de DNS. El cliente puede retransmitir en forma segura estas solicitudes una y otra vez, en caso de que no lleguen respuestas. No importa si el servidor nunca recibió la solicitud o si se perdió la respuesta. Cuando finalmente llegue la respuesta, será la misma (suponiendo que la base de datos de DNS no se actualice en el proceso). Sin embargo, no todas las operaciones son idempotentes porque algunas tienen importantes efectos colaterales; por ejemplo, el incremento a un contador. El uso de RPC para estas operaciones requiere de una semántica más sólida, de modo que cuando el programador llame a un procedimiento, éste no se ejecute varias veces. En este caso tal vez sea necesario establecer una conexión TCP y enviar la solicitud a través de ella, en vez de usar UDP.

6.4.3 Protocolos de transporte en tiempo real

El RPC cliente-servidor es un área en la que UDP se utiliza mucho. Otra área es la de las aplicaciones multimedia en tiempo real. En particular, conforme la radio en Internet, la telefonía en Internet, la música bajo demanda, las videoconferencias, el video bajo demanda y otras aplicaciones multimedia se volvían más comunes, las personas descubrieron que cada una de esas aplicaciones estaba reinventando más o menos el mismo protocolo de transporte de tiempo-real. Cada vez era más claro que tener un protocolo genérico de transporte en tiempo real para múltiples aplicaciones sería una excelente idea.

A raíz de esto fue que nació el **RTP** (**Protocolo de Transporte en Tiempo Real**, del inglés *Real-time Transport Protocol*). Se describe en el RFC 3550 y ahora se utiliza ampliamente para aplicaciones multimedia. A continuación describiremos dos aspectos del transporte en tiempo real. El primero es el protocolo RTP para transportar datos de audio y video en paquetes. El segundo es el procesamiento que se lleva a cabo, en su mayor parte en el receptor, para reproducir el audio y video en el momento correcto. Estas funciones se ajustan a la pila de protocolos según se muestra en la figura 6-30.

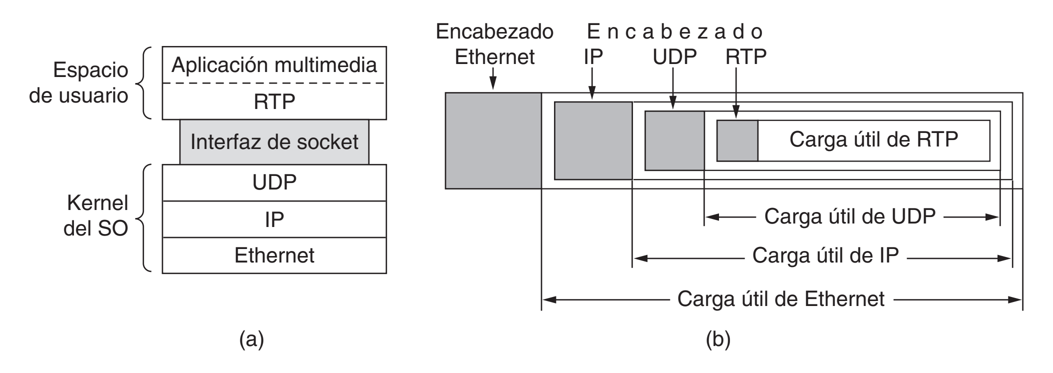


Figura 6-30. (a) La posición de RTP en la pila de protocolos. (b) Anidamiento de paquetes.

Por lo general, RTP se ejecuta en espacio de usuario sobre UDP (en el sistema operativo). Opera como se muestra a continuación. La aplicación multimedia consiste en múltiples flujos de audio, video, texto y quizás otros flujos. Éstos se colocan en la biblioteca RTP, la cual está en el espacio de usuario junto con la aplicación. Esta biblioteca multiplexa los flujos y los codifica en paquetes RTP, que después coloca en un socket. En el extremo del socket correspondiente al sistema operativo, se generan paquetes UDP para envolver los paquetes RTP y se entregan al IP para que los transmita a través de un enlace tal como Ethernet. En el receptor se lleva a cabo el proceso inverso. En un momento dado la aplicación multimedia recibirá datos multimedia de la biblioteca RTP. Es responsable de reproducir los medios. En la figura 6-30(a) se muestra la pila de protocolos para esta situación. En la figura 6-30(b) se muestra el anidamiento de paquetes.

Como consecuencia de este diseño, es un poco difícil saber en cuál capa está RTP. Debido a que se ejecuta en el espacio de usuario y está enlazado al programa de aplicación, sin duda luce como un protocolo de aplicación. Por otro lado, es un protocolo genérico, independiente de la aplicación que simplemente proporciona herramientas de transporte, por lo que se parece también a un protocolo de transporte. Tal vez la mejor descripción sea que es un protocolo de transporte que sólo está implementado en la capa de aplicación, razón por la cual lo estamos analizando en este capítulo.

RTP: el Protocolo de Transporte en Tiempo Real

La función básica de RTP es multiplexar varios flujos de datos de tiempo real en un solo flujo de paquetes UDP. El flujo UDP se puede enviar a un solo destino (unidifusión) o a múltiples destinos (multidifusión). Debido a que RTP sólo utiliza UDP normal, los enrutadores no dan a sus paquetes un trato especial, a menos que se habiliten algunas características de calidad de servicio IP normales. En particular no hay garantías especiales acerca de la entrega, así que los paquetes se pueden perder, retrasar, corromper, etcétera.

El formato de RTP contiene varias características para ayudar a que los receptores trabajen con información multimedia. A cada paquete enviado en un flujo RTP se le da un número más grande que a su predecesor. Esta numeración permite al destino determinar si falta algún paquete en cuyo caso, la mejor acción a realizar queda a criterio de la aplicación. Tal vez esta acción sea omitir una trama de video si los paquetes transportan datos de video, o aproximar el valor faltante mediante la interpolación en caso de que los paquetes transporten datos de audio. La retransmisión no es una opción práctica debido a la probabilidad de que el paquete retransmitido llegue muy tarde como para ser útil. Como consecuencia, RTP no tiene confirmaciones de recepción ni ningún mecanismo para solicitar retransmisiones.

Cada carga útil de RTP podría contener múltiples muestras; éstas se pueden codificar de la forma en la que la aplicación desee. Para permitir la interconectividad, RTP define varios perfiles (por ejemplo, un solo flujo de audio), y para cada perfil se pueden permitir múltiples formatos de codificación. Por ejemplo, un solo flujo de audio se puede codificar como muestras de PCM de 8 bits a 8 kHz mediante codificación delta, codificación predictiva, codificación GSM, MP3, etc. RTP proporciona un campo de encabezado en el que el origen puede especificar la codificación, pero no se involucra de ninguna otra manera en la forma en que se realiza la codificación.

Las estampas de tiempo (*timestamping*) son otra herramienta que muchas de las aplicaciones en tiempo real necesitan. La idea aquí es permitir que la fuente asocie una estampa de tiempo con la primera muestra de cada paquete. Las estampas de tiempo son relativas al inicio del flujo, por lo que sólo son importantes las diferencias entre dichas estampas. Los valores absolutos no tienen significado. Como veremos en breve, este mecanismo permite que el destino haga un uso muy moderado del almacenamiento en búfer y reproduzca cada muestra el número exacto de milisegundos después del inicio del flujo, sin importar cuándo llegó el paquete que contiene la muestra.

La estampa de tiempo no sólo reduce los efectos de la variación en el retardo de la red, sino que también permite que múltiples flujos estén sincronizados entre sí. Por ejemplo, un programa de televisión digital podría tener un flujo de video y dos flujos de audio. Los dos flujos de audio podrían ser para difusiones en estéreo o para manejar películas con la banda sonora del idioma original y con una "traducción" al idioma local, para darle una opción al espectador. Cada flujo proviene de un dispositivo físico diferente, pero si tienen estampas de tiempo de un solo contador, se pueden reproducir de manera síncrona, incluso si los flujos se transmiten o reciben de manera irregular.

En la figura 6-31 se ilustra el encabezado RTP, que consiste de tres palabras de 32 bits y potencialmente de algunas extensiones. La primera palabra contiene el campo *Versión*, que actualmente es la 2. Esperemos que esta versión esté muy cerca de la última debido a que sólo quedó pendiente un punto del código (aunque se puede definir como 3 para indicar que la versión real estaba en una palabra de extensión).

El bit *P* indica que el paquete se ha rellenado para formar un múltiplo de 4 bytes. El último byte de relleno indica cuántos bytes se agregaron. El bit *X* indica que hay un encabezado de extensión. El formato y el significado de este encabezado no se definen. Lo único que se define es que la primera palabra de la extensión proporciona la longitud. Ésta es una puerta de escape para cualquier requerimiento imprevisto.

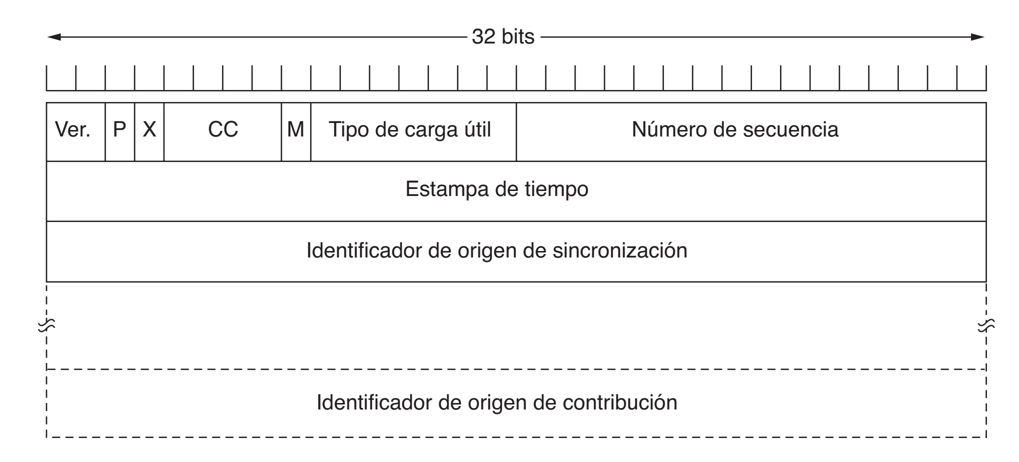


Figura 6-31. El encabezado RTP.

El campo *CC* indica cuántas fuentes de contribución están presentes, de 0 a 15 (vea abajo). El bit *M* es un bit marcador específico de la aplicación. Puede utilizarse para marcar el inicio de una trama de video, el inicio de una palabra en un canal de audio o algo más que la aplicación entienda. El campo *Tipo de carga útil* indica cuál algoritmo de codificación se utilizó (por ejemplo, audio de 8 bits sin compresión, MP3, etc.). Puesto que cada paquete lleva este campo, la codificación puede cambiar durante la transmisión. El *Número de secuencia* es simplemente un contador que se incrementa con cada paquete RTP enviado. Se utiliza para detectar paquetes perdidos.

La fuente del flujo produce la *Estampa de tiempo* para indicar cuándo se creó la primera muestra en el paquete. Este valor puede ayudar a reducir la variabilidad de la sincronización conocida como **variación del retardo** (***jitter***) en el receptor, al desacoplar la reproducción del tiempo de llegada del paquete. El *Identificador de origen de sincronización* indica a cuál flujo pertenece el paquete. Es el método utilizado para multiplexar y desmultiplexar varios flujos de datos en un solo flujo de paquetes UDP. Por último, los *Identificadores de origen de contribución*, en caso de que haya, se utilizan cuando hay mezcladoras en el estudio. En ese caso, la mezcladora es el origen de la sincronización y los flujos que se mezclan se listan aquí.

RTCP: el Protocolo de Control de Transporte en Tiempo Real

RTP tiene un hermano pequeño llamado **RTCP** (**Protocolo de Control de Transporte en Tiempo Real**, del inglés *Real Transport Control Protocol*), el cual se define junto con RTP en el RFC 3550 y se encarga de la retroalimentación, la sincronización y la interfaz de usuario, pero no transporta muestras de medios.

La primera función se puede utilizar para proporcionar a las fuentes retroalimentación sobre el retardo, variación en el retardo o jitter, ancho de banda, congestión y otras propiedades de red. El proceso de codificación puede utilizar esta información para incrementar la tasa de datos (y para proporcionar mejor calidad) cuando la red está funcionando bien y para disminuir la tasa de datos cuando hay problemas en la red. Al proveer una retroalimentación continua, los algoritmos de codificación se pueden adaptar de manera continua para proporcionar la mejor calidad posible bajo las circunstancias actuales. Por ejemplo, si el ancho de banda aumenta o disminuye durante la transmisión, la codificación puede cambiar de MP3 a PCM de 8 bits o a codificación delta, según se requiera. El campo *Tipo de carga útil* se utiliza para indicar al destino cuál algoritmo de codificación se utiliza en el paquete actual, de modo que sea posible modificarlo a solicitud.

Un problema al proveer retroalimentación es que los informes de RTCP se envían a todos los participantes. Para una aplicación de multidifusión con un grupo extenso, el ancho de banda utilizado por RTCP

aumentaría con rapidez. Para evitar que esto ocurra, los emisores de RTCP reducen la tasa de sus informes para que en conjunto no consuman más de, por ejemplo, 5% del ancho de banda de los medios. Para ello, cada participante necesita que el emisor le dé a conocer el ancho de banda de los medios; también necesita conocer el número de participantes, lo cual estima escuchando los otros informes de RTCP.

RTCP también maneja la sincronización entre flujos. El problema es que distintos flujos pueden utilizar relojes diferentes, con distintas granularidades y distintas tasas de derivación. RTCP se puede utilizar para mantenerlos sincronizados.

Por último, RTCP proporciona una forma para nombrar las diferentes fuentes (por ejemplo, en texto ASCII). Esta información se puede desplegar en la pantalla del receptor para indicar quién está hablando en ese momento.

Para encontrar más información sobre RTP, consulte a Perkins (2003).

Reproducción mediante búfer y control de variación de retardo

Una vez que la información de los medios llega al receptor, se debe reproducir en el momento adecuado. En general, éste no será el tiempo en el que llegó el paquete RTP al receptor, ya que los paquetes tardan cantidades un poco distintas de tiempo en transitar por la red. Incluso si los paquetes se inyectan con los intervalos correctos y exactos entre ellos desde el emisor, llegarán al receptor con distintos tiempos relativos. A esta variación en el retardo se le conoce como **jitter**. Incluso una pequeña cantidad de jitter en los paquetes puede provocar molestos artefactos en los medios, como tramas de video entrecortado y audio incomprensible, si los medios simplemente se reproducen a medida que vayan llegando.

La solución a este problema es colocar los paquetes en un **búfer** en el receptor antes de reproducirlos, para reducir el jitter. Como ejemplo, en la figura 6-32 podemos ver que se entrega un flujo de paquetes con una cantidad considerable de jitter. El paquete 1 se envía del servidor en el tiempo $t = 0$ segundos y llega al cliente en el tiempo $t = 1$ segundo. El paquete 2 sufre de un retardo mayor y tarda 2 segundos en llegar. A medida que llegan los paquetes, se colocan en un búfer en la máquina cliente.

Al llegar al tiempo $t = 10$ segundos, empieza la reproducción. En este momento los paquetes 1 a 6 se encuentran en el búfer, por lo que se pueden extraer del búfer a intervalos uniformes para una reproducción continua. En el caso general, no es necesario usar intervalos uniformes debido a que las estampas de tiempo del RTP indican cuándo se deben reproducir los medios.

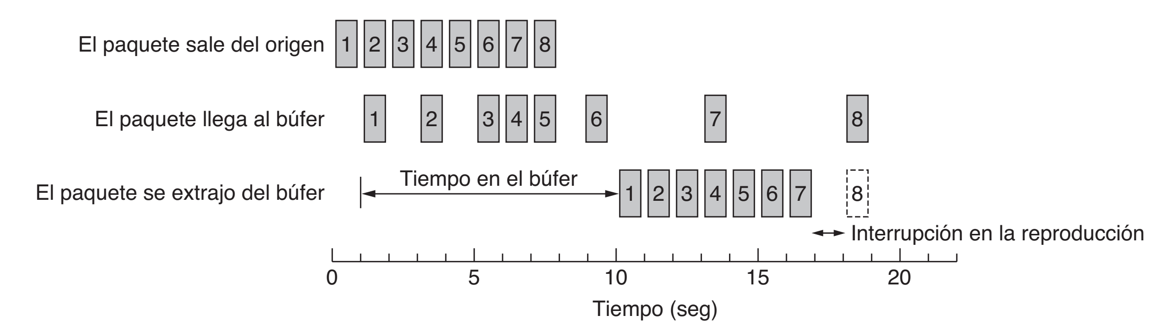


Figura 6-32. Hay que colocar los paquetes en un búfer para uniformar el flujo de salida.

Por desgracia, podemos ver que el paquete 8 se retrasó tanto que no está disponible cuando llega el momento de reproducirlo. Hay dos opciones. Podemos omitir el paquete 8 para que el reproductor avance a los paquetes subsecuentes. O también se puede detener la reproducción hasta que llegue el paquete 8, con lo

cual se crea una molesta interrupción en la música o película. En una aplicación de medios en vivo, como una llamada de voz sobre IP, por lo general se omite el paquete. Las aplicaciones en vivo no funcionan bien con la espera. En una aplicación de medios de flujo continuo el reproductor se podría detener. Podemos aligerar este problema si retardamos aún más el tiempo de inicio, mediante el uso de un búfer más grande. Para un reproductor de audio o video de flujo continuo, a menudo se usan búferes con cerca de 10 segundos para asegurar que el reproductor reciba todos los paquetes (que no se descarten en la red) a tiempo. Para las aplicaciones en vivo, como las videoconferencias, se necesitan búferes cortos para tener una capacidad de respuesta adecuada.

Una consideración clave para tener una reproducción uniforme es el **punto de reproducción**, o qué tanto tiempo esperar los medios en el receptor antes de reproducirlos. La decisión en cuanto al tiempo que debemos esperar depende del jitter. En la figura 6-33 se muestra la diferencia entre una conexión con un jitter bajo y otra con uno alto. El retardo promedio no puede diferir mucho entre las dos, pero si hay un jitter alto, tal vez el punto de reproducción tenga que estar mucho más adelante para poder capturar 99% de los paquetes, en comparación con un jitter bajo.

Para elegir un buen punto de reproducción, la aplicación puede medir el jitter si analiza la diferencia entre las estampas de tiempo RTP y el tiempo de llegada. Cada diferencia proporciona una muestra del retardo (más un desplazamiento fijo arbitrario). Sin embargo, el retardo puede cambiar con el tiempo debido al tráfico adicional que compite por el ancho de banda y a los cambios en las rutas. Para lidiar con este cambio, las aplicaciones pueden adaptar su punto de reproducción mientras se ejecutan. No obstante, si no hacen bien el cambio del punto de reproducción, el usuario puede llegar a percibir un problema técnico. Una manera de evitar este problema en el audio es adaptar el punto de reproducción entre las **ráfagas de voz** (*talkspurts*), en las interrupciones en la conversación. Nadie notará la diferencia entre un silencio corto y uno un poco más largo. RTP permite a las aplicaciones establecer el bit marcador *M* para indicar el inicio de una nueva ráfaga de voz para este fin.

Si el retardo absoluto hasta que se reproduzcan los medios es demasiado largo, las aplicaciones en vivo sufrirán. No se puede hacer nada para reducir el retardo de propagación si ya se está usando una trayectoria directa. Para retraer el punto de reproducción, simplemente hay que aceptar que habrá más paquetes que llegarán demasiado tarde como para reproducirlos. Si esto no es aceptable, la única forma de retraer el punto de reproducción es reducir el jitter mediante el uso de una mejor calidad de servicio; por ejemplo, el servicio diferenciado de reenvío expedito. Es decir, se necesita una mejor red.

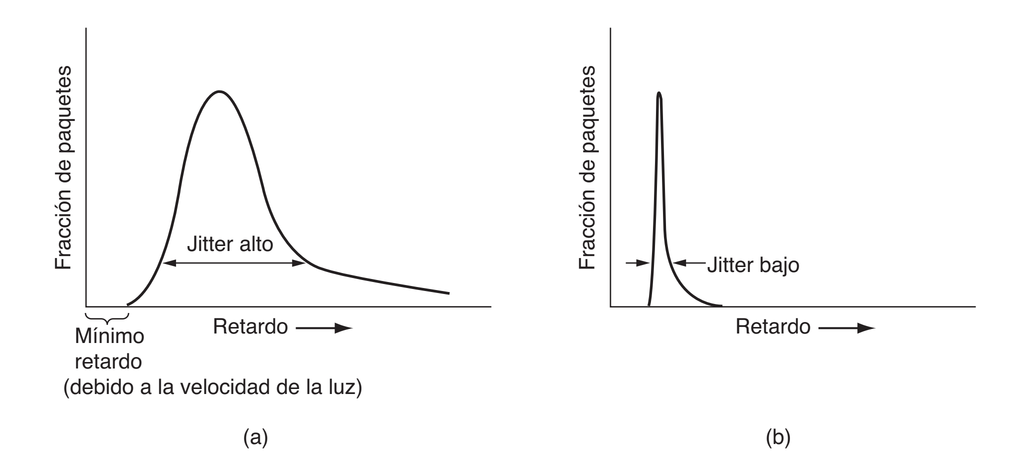


Figura 6-33. (a) Jitter alto. (b) Jitter bajo.

6.5 LOS PROTOCOLOS DE TRANSPORTE DE INTERNET: TCP

UDP es un protocolo simple y tiene algunos usos muy importantes, como las interacciones cliente-servidor y multimedia, pero para la mayoría de las aplicaciones de Internet se necesita una entrega en secuencia confiable. UDP no puede proporcionar esto, por lo que se requiere otro protocolo. Se llama TCP y es el más utilizado en Internet. A continuación lo estudiaremos con detalle.

6.5.1 Introducción a TCP

TCP (**Protocolo de Control de Transmisión**, del inglés *Transmission Control Protocol*) se diseñó específicamente para proporcionar un flujo de bytes confiable de extremo a extremo a través de una interred no confiable. Una interred difiere de una sola red debido a que sus diversas partes podrían tener diferentes topologías, anchos de banda, retardos, tamaños de paquete y otros parámetros. TCP se diseñó para adaptarse de manera dinámica a las propiedades de la interred y sobreponerse a muchos tipos de fallas.

TCP se definió formalmente en el RFC 793 en septiembre de 1981. Con el paso del tiempo se han realizado muchas mejoras y se han corregido varios errores e inconsistencias. Para que el lector tenga una idea de la magnitud de TCP, los RFC importantes son ahora el RFC 793 más: las aclaraciones y correcciones de errores en el RFC 1122; las extensiones para un alto desempeño en el RFC 1323; las confirmaciones de recepción selectivas en el RFC 2018; el control de congestión en el RFC 2581; la readaptación de los campos del encabezado para la calidad del servicio en el RFC 2873; los temporizadores de retransmisión mejorados en el RFC 2988 y la notificación explícita de congestión en el RFC 3168. La colección completa es todavía más grande, por lo cual se produjo una guía para los diversos documentos RFC, que por supuesto se publicó como otro documento RFC: el RFC 4614.

Cada máquina que soporta TCP tiene una entidad de transporte TCP, ya sea un procedimiento de biblioteca, un proceso de usuario o (lo más común) sea parte del kernel. En todos los casos, maneja flujos TCP e interactúa con la capa IP. Una entidad TCP acepta flujos de datos de usuario de procesos locales, los divide en fragmentos que no excedan los 64 KB (en la práctica, por lo general son 1460 bytes de datos para ajustarlos en una sola trama Ethernet con los encabezados IP y TCP), y envía cada pieza como un datagrama IP independiente. Cuando los datagramas que contienen datos TCP llegan a una máquina, se pasan a la entidad TCP, la cual reconstruye los flujos de bytes originales. Con el afán de simplificar, algunas veces utilizaremos sólo "TCP" para referirnos a la entidad de transporte TCP (una pieza de software) o al protocolo TCP (un conjunto de reglas). El contexto dejará claro a que nos referimos. Por ejemplo, en la frase "El usuario proporciona los datos a TCP", es claro que nos referimos a la entidad de transporte TCP.

La capa IP no ofrece ninguna garantía de que los datagramas se entregarán de manera apropiada, ni tampoco una indicación sobre qué tan rápido se pueden enviar los datagramas. Corresponde a TCP enviar los datagramas con la suficiente rapidez como para hacer uso de la capacidad sin provocar una congestión; también le corresponde terminar los temporizadores y retransmitir los datagramas que no se entreguen. Los datagramas que sí lleguen podrían hacerlo en el orden incorrecto; también corresponde a TCP reensamblarlos en mensajes con la secuencia apropiada. En resumen, TCP debe proporcionar un buen desempeño con la confiabilidad que la mayoría de las aplicaciones desean y que IP no proporciona.

6.5.2 El modelo del servicio TCP

El servicio TCP se obtiene al hacer que tanto el servidor como el receptor creen puntos terminales, llamados **sockets**, como se mencionó en la sección 6.1.3. Cada socket tiene un número (dirección) que consiste en la dirección IP del host y un número de 16 bits que es local para ese host, llamado **puerto**. Un puerto

es el nombre TCP para un TSAP. Para obtener el servicio TCP, hay que establecer de manera explícita una conexión entre un socket en una máquina y un socket en otra máquina. Las llamadas de socket se listan en la figura 6-5.

Podemos usar un socket para múltiples conexiones al mismo tiempo. En otras palabras, dos o más conexiones pueden terminar en el mismo socket. Las conexiones se identifican mediante los identificadores de socket de los dos extremos; esto es, (*socket1*, *socket2*). No se utilizan números de circuitos virtuales u otros identificadores.

Los números de puerto menores que 1024 están reservados para los servicios estándar que, por lo general, sólo los usuarios privilegiados pueden iniciar (por ejemplo, el usuario root en los sistemas UNIX). Éstos se llaman **puertos bien conocidos**. Por ejemplo, cualquier proceso que desee recuperar en forma remota el correo de un host se puede conectar con el puerto 143 del host de destino para contactarse con su demonio (daemon) IMAP. La lista de puertos bien conocidos se proporciona en www.iana.org. Se han asignado más de 700. En la figura 6-34 se listan algunos de los más conocidos.

Puerto	Protocolo	Uso
20, 21	FTP	Transferencia de archivos.
22	SSH	Inicio de sesión remoto, reemplazo de Telnet.
25	SMTP	Correo electrónico.
80	HTTP	World Wide Web.
110	POP-3	Acceso remoto al correo electrónico.
143	IMAP	Acceso remoto al correo electrónico.
443	HTTPS	Acceso seguro a web (HTTP sobre SSL/TLS).
543	RTSP	Control del reproductor de medios.
631	IPP	Compartición de impresoras.

Figura 6-34. Algunos puertos asignados.

Se pueden registrar otros puertos del 1024 hasta el 49151 con la IANA para que los usuarios sin privilegios puedan usarlos, pero las aplicaciones pueden y seleccionan sus propios puertos. Por ejemplo, la aplicación BitTorrent para compartir archivos de igual a igual usa (de manera informal) los puertos 6881 a 6887, pero también puede operar en otros puertos.

Sin duda podría ser posible que el demonio FTP se conecte por sí solo al puerto 21 en tiempo de arranque, que el demonio SSH se conecte por sí solo al puerto 22 en tiempo de arranque, y así en lo sucesivo. Sin embargo, hacer lo anterior podría llenar la memoria con demonios que están inactivos la mayor parte del tiempo. En su lugar, lo que se hace por lo general es que un solo demonio, llamado *demonio de Internet* (*inetd*) en UNIX, se conecte por sí solo a múltiples puertos y espere la primera conexión entrante. Cuando eso ocurre, *inetd* bifurca un nuevo proceso y ejecuta el demonio apropiado en él, para dejar que ese demonio maneje la solicitud. De esta forma, los demonios distintos a *inetd* sólo están activos cuando hay trabajo para ellos. Inetd consulta un archivo de configuración para saber cuál puerto utilizar. En consecuencia, el administrador del sistema puede configurar el sistema para tener demonios permanentes en los puertos más ocupados (por ejemplo, el puerto 80) e *inetd* en los demás.

Todas las conexiones TCP son *full dúplex* y de punto a punto. *Full dúplex* significa que el tráfico puede ir en ambas direcciones al mismo tiempo. Punto a punto significa que cada conexión tiene exactamente dos puntos terminales. TCP no soporta la multidifusión ni la difusión.

Una conexión TCP es un flujo de bytes, no un flujo de mensajes. Los límites de los mensajes no se preservan de un extremo a otro. Por ejemplo, si el proceso emisor realiza cuatro escrituras de 512 bytes en un flujo TCP, tal vez estos datos se entreguen al proceso receptor como cuatro fragmentos de 512 bytes, dos fragmentos de 1024 bytes, uno de 2 048 bytes (vea la figura 6-35), o de alguna otra forma. No hay manera de que el receptor detecte la(s) unidad(es) en la(s) que se escribieron los datos, sin importar qué tanto se esfuerce.

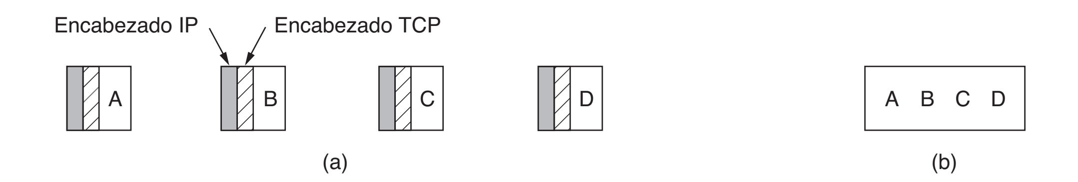


Figura 6-35. (a) Cuatro segmentos de 512 bytes que se envían como diagramas IP separados. (b) Los 2 048 bytes de datos que se entregan a la aplicación en una sola llamada READ.

Los archivos de UNIX también tienen esta propiedad. El lector de un archivo no puede indicar si éste se escribió un bloque a la vez, un byte a la vez o todo al mismo tiempo. Al igual que con un archivo de UNIX, el software TCP no tiene idea de lo que significan los bytes y no le interesa averiguarlo. Un byte es sólo un byte.

Cuando una aplicación pasa datos a TCP, éste decide entre enviarlos de inmediato o almacenarlos en el búfer (con el fin de recolectar una mayor cantidad y enviar todos los datos al mismo tiempo). Sin embargo, algunas veces la aplicación en realidad necesita que los datos se envíen de inmediato. Por ejemplo, suponga que el usuario de un juego interactivo quiere enviar un flujo continuo de actualizaciones. Es esencial que éstas se envíen de inmediato y no se almacenen en el búfer hasta que haya toda una colección. Para forzar la salida de los datos, TCP tiene la noción de una bandera PUSH que se transporta en los paquetes. La intención original era permitir que las aplicaciones indiquen a las implementaciones de TCP a través de la bandera PUSH que no retarden la transmisión. Sin embargo, las aplicaciones no pueden establecer literalmente la bandera PUSH cuando envían datos. En cambio, los distintos sistemas operativos han desarrollado distintas opciones para agilizar la transmisión (por ejemplo, TCP_NODELAY en Windows y Linux).

Para los arqueólogos de Internet, también mencionaremos una característica interesante del servicio TCP que permanece en el protocolo pero se usa muy raras veces: **datos urgentes**. Cuando una aplicación tiene datos de alta prioridad que se deben procesar de inmediato, por ejemplo cuando un usuario interactivo oprime las teclas CTRL-C para interrumpir un cálculo remoto que está en proceso, la aplicación emisora puede colocar cierta información de control en el flujo de datos y entregarla a TCP junto con la bandera URGENT. Este evento hace que TCP deje de acumular datos y transmita de inmediato todo lo que tiene para esa conexión.

Cuando los datos urgentes se reciben en el destino, la aplicación receptora se interrumpe (es decir, recibe una señal en términos de UNIX) para poder lo que estaba haciendo y leer el flujo de datos para encontrar los datos urgentes. El final de los datos urgentes se marca de modo que la aplicación sepa cuándo terminan. El inicio de los datos urgentes no se marca. Depende de la aplicación averiguarlo.

Este esquema proporciona básicamente un mecanismo de señalización simple y deja todo lo demás a la aplicación. Sin embargo, aunque los datos urgentes son útiles en potencia, no se encontró ninguna apli-

cación convincente para ellos desde un principio y dejaron de usarse. En la actualidad no se recomienda su uso debido a las diferencias de implementación, por lo que las aplicaciones tienen que manejar su propia señalización. Tal vez los futuros protocolos de transporte provean una mejor señalización.

6.5.3 El protocolo TCP

En esta sección daremos un repaso general del protocolo TCP. En la siguiente veremos el encabezado del protocolo, campo por campo.

Una característica clave de TCP, y una que domina el diseño del protocolo, es que cada byte de una conexión TCP tiene su propio número de secuencia de 32 bits. Cuando Internet comenzó, las líneas entre los enrutadores eran principalmente líneas rentadas de 56 kbps, por lo que un host que mandaba datos a toda velocidad tardaba una semana en recorrer los números de secuencia. A las velocidades de las redes modernas, los números de secuencia se pueden consumir con una rapidez alarmante, como veremos más adelante. Los números de secuencia separados de 32 bits se transmiten en paquetes para la posición de ventana deslizante en una dirección, y para las confirmaciones de recepción en la dirección opuesta, como veremos a continuación.

La entidad TCP emisora y receptora intercambian datos en forma de segmentos. Un **segmento TCP** consiste en un encabezado fijo de 20 bytes (más una parte opcional), seguido de cero o más bytes de datos. El software de TCP decide qué tan grandes deben ser los segmentos. Puede acumular datos de varias escrituras para formar un segmento, o dividir los datos de una escritura en varios segmentos. Hay dos límites que restringen el tamaño de segmento. Primero, cada segmento, incluido el encabezado TCP, debe caber en la carga útil de 65 515 bytes del IP. Segundo, cada enlace tiene una **MTU** (**Unidad Máxima de Transferencia**, del inglés *Maximum Transfer Unit*). Cada segmento debe caber en la MTU en el emisor y el receptor, de modo que se pueda enviar y recibir en un solo paquete sin fragmentar. En la práctica, la MTU es por lo general de 1500 bytes (el tamaño de la carga útil en Ethernet) y, por tanto, define el límite superior en el tamaño de segmento.

Sin embargo, de todas formas es posible que los paquetes IP que transportan segmentos TCP se fragmenten al pasar por una trayectoria de red en la que algún enlace tenga una MTU pequeña. Si esto ocurre, degradará el desempeño y provocará otros problemas (Kent y Mogul, 1987). En cambio, las implementaciones modernas de TCP realizan el **descubrimiento de MTU de la ruta** mediante el uso de la técnica descrita en el RF 1191 y que describimos en la sección 5.5.5. Esta técnica usa mensajes de error de ICMP para encontrar la MTU más pequeña para cualquier enlace en la ruta. Después, TCP ajusta el tamaño del segmento en forma descendente para evitar la fragmentación.

El protocolo básico que utilizan las entidades TCP es el protocolo de ventana deslizante con un tamaño dinámico de ventana. Cuando un emisor transmite un segmento, también inicia un temporizador. Cuando llega el segmento al destino, la entidad TCP receptora devuelve un segmento (con datos si existen, de otro modo sin ellos) que contiene un número de confirmación de recepción igual al siguiente número de secuencia que espera recibir, junto con el tamaño de la ventana remanente. Si el temporizador del emisor expira antes de recibir la confirmación de recepción, el emisor transmite de nuevo el segmento.

Aunque este protocolo suena sencillo, tiene muchos detalles prácticos que algunas veces son sutiles, los cuales explicaremos a continuación. Los segmentos pueden llegar fuera de orden, por lo que los bytes 3 072-4 095 pueden llegar pero tal vez sin que se envíe su respectiva confirmación de recepción, porque los bytes 2 048-3 071 no han aparecido aún. Además, los segmentos se pueden retardar tanto tiempo en tránsito que el temporizador del emisor expirará y este último retransmitirá los segmentos. Las retransmisiones podrían incluir rangos de bytes diferentes a los de la transmisión original, por lo cual se requiere una administración cuidadosa para llevar el control de los bytes que se han recibido de manera correcta en un momento determinado. Sin embargo, esto es factible ya que cada byte del flujo tiene su propio desplazamiento único.

El TCP debe estar preparado para manejar y resolver estos problemas de una manera eficiente. Se ha invertido una cantidad considerable de esfuerzo en la optimización del desempeño de los flujos TCP, incluso frente a problemas de red. A continuación estudiaremos varios de los algoritmos usados por muchas implementaciones de TCP.

6.5.4 El encabezado del segmento TCP

En la figura 6-36 se muestra la distribución de un segmento TCP. Cada segmento comienza con un encabezado de formato fijo de 20 bytes. El encabezado fijo puede ir seguido de encabezado de opciones. Después de las opciones, si las hay, pueden continuar hasta 65 535 − 20 − 20 = 65 495 bytes de datos, donde los primeros 20 se refieren al encabezado IP y los segundos al encabezado TCP. Los segmentos sin datos son legales y se usan por lo común para confirmaciones de recepción y mensajes de control.

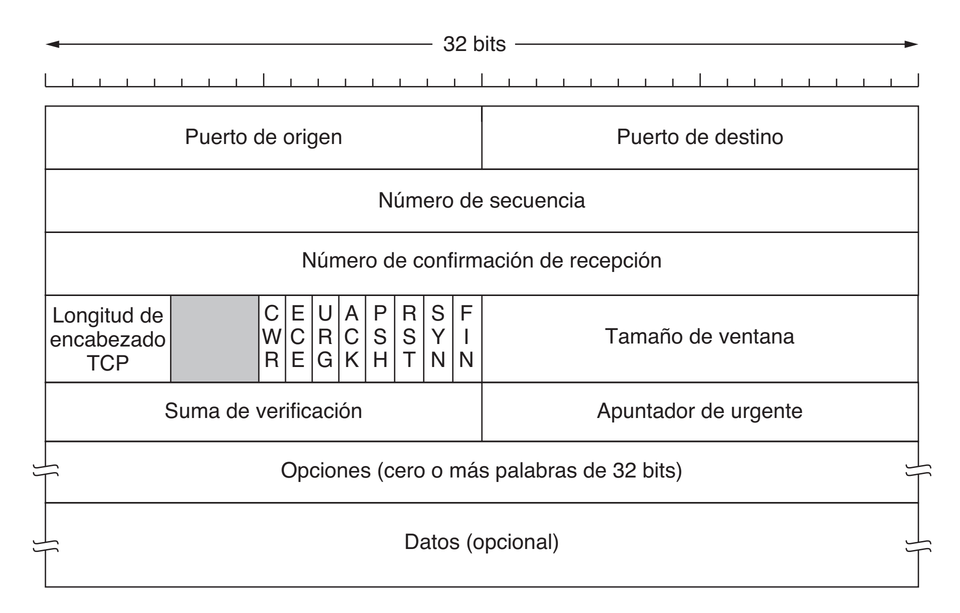


Figura 6-36. El encabezado TCP.

Ahora examinaremos el encabezado TCP campo por campo. Los campos *Puerto de origen* y *Puerto de destino* identifican los puntos terminales locales de la conexión. Un puerto TCP más la dirección IP de su host forman un punto terminal único de 48 bits. Los puntos terminales de origen y de destino en conjunto identifican la conexión. Este identificador de conexión se denomina **5-tupla,** ya que consiste en cinco piezas de información: el protocolo (TCP), IP de origen y puerto de origen, IP de destino y puerto de destino.

Los campos *Número de secuencia* y *Número de confirmación de recepción* desempeñan sus funciones normales. Cabe mencionar que el segundo especifica el siguiente byte en el orden esperado, no el último byte de manera correcta recibido. Es una **confirmación de recepción acumulativa** debido a que sintetiza los datos recibidos con un solo número. No va más allá de los datos perdidos. Ambos tienen 32 bits de longitud porque cada byte de datos está numerado en un flujo TCP.

La *Longitud del encabezado TCP* indica la cantidad de palabras de 32 bits contenidas en el encabezado TCP. Esta información es necesaria porque el campo *Opciones* es de longitud variable por lo que el encabezado también. Técnicamente, este campo en realidad indica el comienzo de los datos en el segmento,

medido en palabras de 32 bits, pero ese número es simplemente la longitud del encabezado en palabras, por lo que el efecto es el mismo.

A continuación viene un campo de 4 bits que no se usa. El hecho de que estos bits hayan permanecido sin ser utilizados durante 30 años (pues sólo se han reclamado 2 de los 6 bits reservados en un inicio) es testimonio de lo bien pensado que está l TCP. Protocolos inferiores habrían necesitado estos bits para corregir errores en el diseño original.

A continuación vienen ocho banderas de 1 bit. *CWR* y *ECE* se utilizan para indicar congestión cuando se usa ECN (Notificación Explícita de Congestión), como se especifica en el RFC 3168. *ECE* se establece para indicar una *Eco de ECN* (*ECN-Echo*) a un emisor TCP y decirle que reduzca su velocidad cuando el receptor TCP recibe una indicación de congestión de la red. *CWR* se establece para indicar una *Ventana de congestión reducida* del emisor TCP al receptor TCP, de modo que sepa que el emisor redujo su velocidad y puede dejar de enviar la *Repetición de ECN*. En la sección 6.5.10 analizaremos el papel de ECN en el control de congestión de TCP.

URG se establece en 1 si está en uso el *Apuntador urgente*. El *Apuntador urgente* se usa para indicar un desplazamiento en bytes a partir del número de secuencia actual en el que se deben encontrar los datos urgentes. Este recurso sustituye los mensajes de interrupción. Como se mencionó antes, este recurso es un mecanismo rudimentario para permitir que el emisor envíe una señal al receptor sin implicar a TCP en razón de la interrupción, pero se usa muy poco.

El bit *ACK* se establece en 1 para indicar que el *Número de confirmación de recepción* es válido. Éste es el caso para casi todos los paquetes. Si *ACK* es 0, el segmento no contiene una confirmación de recepción, por lo que se ignora el campo *Número de confirmación de recepción*.

El bit *PSH* indica datos que se deben transmitir de inmediato (*PUSHed data*). Por este medio se solicita atentamente al receptor que entregue los datos a la aplicación a su llegada y no los almacene en búfer hasta que se haya recibido un búfer completo (lo que podría hacer en otras circunstancias por razones de eficiencia).

El bit *RST* se usa para restablecer de manera repentina una conexión que se ha confundido debido a una falla de host o alguna otra razón. También se usa para rechazar un segmento no válido o un intento de abrir una conexión. Por lo general, si usted recibe un segmento con el bit *RST* encendido, tiene un problema entre manos.

El bit *SYN* se usa para establecer conexiones. La solicitud de conexión tiene *SYN* = 1 y *ACK* = 0 para indicar que el campo de confirmación de recepción superpuesto no está en uso. Pero la respuesta de conexión sí lleva una confirmación de recepción, por lo que tiene *SYN* = 1 y *ACK* = 1. En esencia, el bit *SYN* se usa para denotar CONNECTION REQUEST y CONNECTION ACCEPTED, y el bit *ACK* sirve para distinguir entre ambas posibilidades.

El bit *FIN* se usa para liberar una conexión y especifica que el emisor no tiene más datos que *transmitir*. Sin embargo, después de cerrar una conexión, el proceso encargado del cierre puede continuar *recibiendo* datos de manera indefinida. Ambos segmentos *SYN* y *FIN* tienen números de secuencia y, por tanto, se garantiza su procesamiento en el orden correcto.

El control de flujo en TCP se maneja mediante una ventana deslizante de tamaño variable. El campo *Tamaño de ventana* indica la cantidad de bytes que se pueden enviar, comenzando por el byte cuya recepción se ha confirmado. Un campo de *Tamaño de ventana* de 0 es válido e indica que se han recibido los bytes hasta *Número de confirmación de recepción* − 1, inclusive, pero que el receptor no ha tenido oportunidad de consumir los datos y ya no desea más datos por el momento. El receptor puede otorgar permiso más tarde para enviar mediante la transmisión de un segmento con el mismo *Número de confirmación de recepción* y un campo *Tamaño de ventana* distinto de cero.

En los protocolos del capítulo 3, las confirmaciones de recepción de las tramas recibidas y los permisos para enviar nuevas tramas estaban enlazados. Ésta fue una consecuencia de un tamaño de ventana fijo para cada protocolo. En TCP, las confirmaciones de recepción y los permisos para enviar datos adiciona-

les son por completo independientes. En efecto, un receptor puede decir: “He recibido bytes hasta *k*, pero por ahora no deseo más”. Esta independencia (de hecho, una ventana de tamaño variable) proporciona una flexibilidad adicional. A continuación lo estudiaremos con más detalle.

También se proporciona una *Suma de verificación* para agregar confiabilidad. Realiza una suma de verificación del encabezado, los datos y un pseudoencabezado conceptual exactamente de la misma forma que UDP, sólo que el pseudoencabezado tiene el número de protocolo para TCP (6) y la suma de verificación es obligatoria. Consulte la sección 6.4.1 si desea más detalles.

El campo *Opciones* ofrece una forma de agregar las características adicionales que no están cubiertas por el encabezado normal. Se han definido muchas opciones, de las cuales varias se usan con frecuencia. Las opciones son de longitud variable, llenan un múltiplo de 32 bits mediante la técnica de relleno con ceros y se pueden extender hasta 40 bytes para dar cabida al encabezado TCP más largo que se pueda especificar. Algunas opciones se transmiten cuando se establece una conexión para negociar o informar al otro extremo sobre las capacidades disponibles. Otras opciones se transmiten en paquetes durante el tiempo de vida de la conexión. Cada opción tiene una codificación Tipo-Longitud-Valor (TLV).

Una opción muy utilizada es la que permite a cada host especificar el **MSS** (**Tamaño Máximo de Segmento**, del inglés *Maximum Segment Size*) que está dispuesto a aceptar. Es más eficiente usar segmentos grandes que segmentos pequeños, debido a que el encabezado de 20 bytes se puede amortizar entre más datos sean, pero los hosts pequeños tal vez no puedan manejar segmentos muy grandes. Durante el establecimiento de la conexión, cada lado puede anunciar su máximo y ver el de su compañero. Si un host no usa esta opción, tiene una carga útil predeterminada de 536 bytes. Se requiere que todos los hosts de Internet acepten segmentos TCP de 536 + 20 = 556 bytes. No es necesario que el tamaño máximo de segmento en ambas direcciones sea el mismo.

En las líneas con gran ancho de banda, alto retardo o ambas cosas, la ventana de 64 KB que corresponde a un campo de 16 bits es un problema. Por ejemplo, en una línea OC-12 (de al rededor de 600 Mbps), se requiere menos de 1 mseg para enviar una ventana de 64 KB completa. Si el retardo de propagación de ida y vuelta es de 50 mseg (lo cual es común para la fibra óptica transcontinental), el emisor estará inactivo por más de 98% del tiempo en espera de confirmaciones de recepción. Un tamaño de ventana más grande permitiría al emisor continuar enviando datos. La opción **escala de ventana** permite al emisor y al receptor negociar un factor de escala de ventana al inicio de una conexión. Ambos lados utilizan el factor de escala para desplazar el campo *Tamaño de ventana* hasta un máximo de 14 bits a la izquierda, con lo cual se permiten ventanas de hasta 2^{30} bytes. La mayoría de las implementaciones de TCP soportan esta opción.

La opción **estampa de tiempo** transmite una estampa de tiempo enviada por el emisor y repetida por el receptor. Se incluye en todos los paquetes una vez que se define su uso durante el establecimiento de la conexión, y se utiliza para calcular muestras del tiempo de ida y vuelta que se utilizan para estimar cuando se ha perdido un paquete. También se utiliza como extensión lógica del número de secuencia de 32 bits. En una conexión rápida, el número de secuencia se puede reiniciar muy rápido, lo cual puede llegar a provocar una confusión entre los datos viejos y los nuevos. El esquema **PAWS** (**Protección Contra el Reinicio de Números de Secuencia**, del inglés *Protection Against Wrapped Sequence numbers*) descarta los segmentos entrantes que tengan estampas de tiempo viejas para evitar este problema.

Por último, la opción **SACK** (**Confirmación de Recepción Selectiva**, del inglés *Selective ACKnowledgement*) permite a un receptor indicar al emisor los rangos de números de secuencia que ha recibido. Complementa el *Número de confirmación de recepción* y se utiliza después de haber perdido un paquete y de la llegada de los datos subsiguientes (o duplicados). Los nuevos datos no se reflejan mediante el campo *Número de confirmación de recepción* en el encabezado debido a que ese campo sólo proporciona el siguiente byte en el orden que se espera. Con SACK, el emisor está explícitamente consciente de los datos que tiene el receptor y, por ende, puede determinar qué datos se deben retransmitir. SACK se define

en el RFC 2108 y en el RFC 2883; su uso es cada vez más frecuente. En la sección 6.5.10 describiremos el uso de SACK junto con el control de congestión.

6.5.5 Establecimiento de una conexión TCP

En TCP las conexiones se establecen mediante el acuerdo de tres vías que estudiamos en la sección 6.2.2. Para establecer una conexión, uno de los lados (digamos que el servidor) espera en forma pasiva una conexión entrante mediante la ejecución de las primitivas LISTEN y ACCEPT en ese orden, ya sea que se especifique un origen determinado o a nadie en particular.

El otro lado (digamos que el cliente) ejecuta una primitiva CONNECT en la que especifica la dirección y el puerto con el que se desea conectar, el tamaño máximo de segmento TCP que está dispuesto a aceptar y de manera opcional algunos datos de usuario (por ejemplo, una contraseña). La primitiva CONNECT envía un segmento TCP con el bit *SYN* encendido y el bit *ACK* apagado, y espera una respuesta.

Cuando este segmento llega al destino, la entidad TCP de ahí revisa si hay un proceso que haya ejecutado una primitiva LISTEN en el puerto que se indica en el campo *Puerto de destino*. Si no lo hay, envía una respuesta con el bit *RST* encendido para rechazar la conexión.

Si algún proceso está escuchando en el puerto, ese proceso recibe el segmento TCP entrante y puede entonces aceptar o rechazar la conexión. Si la acepta, se devuelve un segmento de confirmación de recepción. La secuencia de segmentos TCP enviados en el caso normal se muestra en la figura 6-37(a). Observe que un segmento *SYN* consume 1 byte de espacio de secuencia, por lo que se puede reconocer sin ambigüedades.

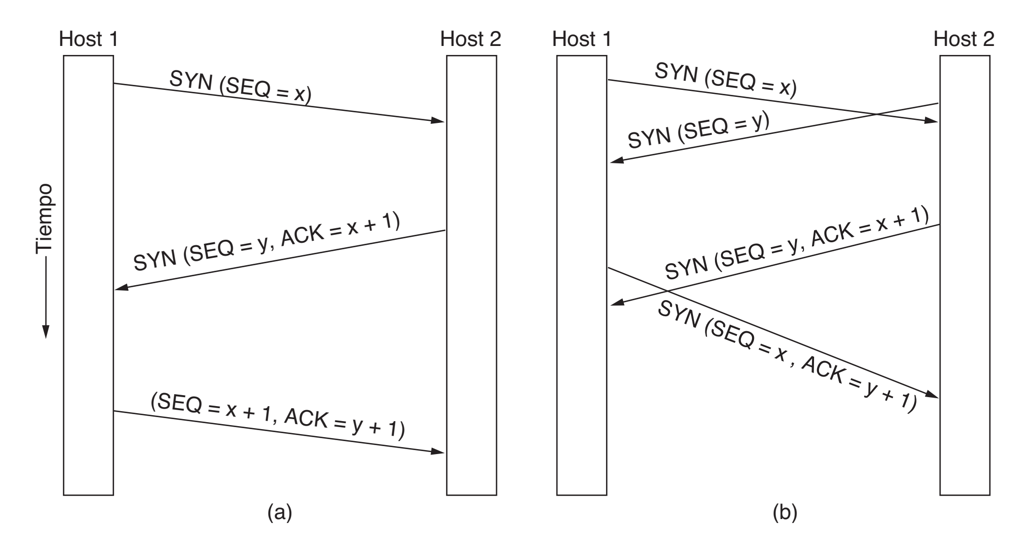


Figura 6-37. (a) Establecimiento de una conexión TCP en el caso normal. (b) Establecimiento de una conexión simultánea en ambos lados.

En el caso en que dos hosts intenten establecer al mismo tiempo una conexión entre los mismos dos sockets, la secuencia de eventos es como se ilustra en la figura 6-37(b). El resultado de estos eventos es que sólo se establece una conexión y no dos, pues las conexiones se identifican por sus puntos terminales. Si el primer establecimiento resulta en una conexión identificada por (x, y) y el segundo también, sólo se hace una entrada de tabla; a saber, para (x, y).

Recuerde que el número de secuencia inicial elegido por cada host se debe reiniciar con lentitud en vez de ser una constante tal como 0. Esta regla es para protegerse contra los paquetes duplicados retardados, como vimos en la sección 6.2.2. En un principio esto se lograba mediante un esquema basado en reloj, en donde éste emitía un pulso cada 4 μseg.

Sin embargo, una vulnerabilidad al implementar el acuerdo de tres vías es que el proceso de escucha debe recordar su número de secuencia tan pronto como responde con su propio segmento *SYN*. Esto significa que un emisor malicioso puede ocupar los recursos en un host si envía un flujo continuo de segmentos *SYN* y nunca les da seguimiento para completar la conexión. A este ataque se le conoce como **inundación SYN** y logró inutilizar varios servidores web en la década de 1990.

Una manera de defenderse contra este ataque es mediante el uso de la técnica **SYN cookies**. En vez de recordar el número de secuencia, un host selecciona un número de secuencia generado en forma criptográfica, lo coloca en el segmento de salida y se olvida de él. Si se completa el acuerdo de tres vías, este número de secuencia (más 1) se devolverá al host. Así, para regenerar el número de secuencia correcto hay que ejecutar la misma función criptográfica, siempre y cuando se conozcan las entradas para esa función; por ejemplo, la dirección IP y puerto del otro host, además de un secreto local. Este procedimiento permite al host verificar que un número de secuencia cuya recepción esté confirmada sea correcto sin tener que recordar el número de secuencia por separado. Existen algunos riesgos, como la incapacidad de manejar opciones TCP, por lo que la técnica SYN cookies sólo se puede usar cuando el host es sujeto de una inundación SYN. Sin embargo, son una interesante variante con relación al establecimiento de conexiones. Para obtener más información, consulte el RFC 4987 y a Lemon (2002).

6.5.6 Liberación de una conexión TCP

Aunque las conexiones TCP son *full dúplex*, para entender la manera en que se liberan las conexiones es mejor visualizarlas como un par de conexiones simplex. Cada conexión simplex se libera de manera independiente de su igual. Para liberar una conexión, cualquiera de las partes puede enviar un segmento TCP con el bit *FIN* establecido, lo que significa que no tiene más datos por transmitir. Al confirmarse la recepción de *FIN*, se apaga ese sentido para que no se transmitan nuevos datos. Sin embargo, lo datos pueden seguir fluyendo de manera indefinida por el otro sentido. Cuando se apagan ambos sentidos, se libera la conexión. Por lo general se requieren cuatro segmentos TCP para liberar una conexión: un *FIN* y un *ACK* para cada sentido. Sin embargo, es posible que el primer *ACK* y el segundo *FIN* estén contenidos en el mismo segmento, con lo cual se reduce la cuenta total a tres.

Al igual que con las llamadas telefónicas en las que ambas partes dicen adiós y cuelgan el teléfono al mismo tiempo, ambos extremos de una conexión TCP pueden enviar segmentos *FIN* al mismo tiempo. La recepción de ambos se confirma de la manera usual, y se apaga la conexión. De hecho, en esencia no hay diferencia entre la liberación secuencial o simultánea por parte de los hosts.

Para evitar el problema de los dos ejércitos (que vimos en la sección 6.2.3), se usan temporizadores. Si no llega una respuesta a un *FIN* en un máximo de dos tiempos de vida del paquete, el emisor del *FIN* libera la conexión. Tarde o temprano el otro lado notará que, al parecer, ya nadie lo está escuchando, y también expirará su temporizador. Aunque esta solución no es perfecta, dado el hecho de que teóricamente es imposible una solución perfecta, tendremos que conformarnos con ella. En la práctica, pocas veces ocurren problemas.

6.5.7 Modelado de administración de conexiones TCP

Los pasos requeridos para establecer y liberar conexiones se pueden representar en una máquina de estados finitos con los 11 estados que se listan en la figura 6-38. En cada estado son legales ciertos eventos. Al ocurrir un evento legal, debe emprenderse alguna acción. Si ocurre algún otro evento, se reporta un error.

Cada conexión comienza en el estado CLOSED (cerrado) y deja ese estado cuando hace una apertura pasiva (LISTEN) o una apertura activa (CONNECT). Si el otro lado realiza la acción opuesta, se establece una conexión y el estado se vuelve ESTABLISHED (establecida). La liberación de la conexión se puede iniciar desde cualquiera de los dos lados. Al completarse, el estado regresa a CLOSED.

Estado	Descripción
CLOSED	No hay conexión activa o pendiente.
LISTEN	El servidor espera una llamada entrante.
SYN RCVD	Llegó una solicitud de conexión; espera ACK.
SYN SENT	La aplicación empezó a abrir una conexión.
ESTABLISHED	El estado normal de transferencia de datos.
FIN WAIT 1	La aplicación notificó que ya terminó.
FIN WAIT 2	El otro lado acordó liberar.
TIME WAIT	Espera a que todos los paquetes expiren.
CLOSING	Ambos lados intentaron cerrar al mismo tiempo.
CLOSE WAIT	El otro lado inició una liberación.
LAST ACK	Espera a que todos los paquetes expiren.

Figura 6-38. Los estados utilizados en la máquina de estados finitos para administrar conexiones TCP.

La máquina de estados finitos se muestra en la figura 6-39. El caso común de un cliente que se conecta activamente a un servidor pasivo se indica con líneas gruesas (continuas para el cliente, punteadas para el servidor). Las líneas delgadas son secuencias de eventos no usuales. Cada línea de la figura 6-39 se marca mediante un par *evento/acción*. El evento puede ser una llamada de sistema iniciada por el usuario (CONNECT, LISTEN, SEND o CLOSE), la llegada de un segmento (*SYN*, *FIN*, *ACK* o *RST*) o, en un caso, una expiración de temporizador del doble del tiempo de vida máximo del paquete. La acción es el envío de un segmento de control (*SYN*, *FIN* o *RST*) o nada, lo cual se indica mediante (—). Los comentarios aparecen entre paréntesis.

Podemos entender mejor el diagrama si seguimos primero la trayectoria de un cliente (la línea continua gruesa) y luego la de un servidor (línea punteada gruesa). Cuando un programa de aplicación en la máquina cliente emite una solicitud CONNECT, la entidad TCP local crea un registro de conexión, lo marca para indicar que está en el estado *SYN SENT* y envía un segmento *SYN*. Cabe mencionar que muchas conexiones pueden estar abiertas (o en proceso de apertura) al mismo tiempo como parte de varias aplicaciones, por lo que el estado es por conexión y se graba en el registro de conexiones. Al llegar el *SYN* + *ACK*, TCP envía el último *ACK* del acuerdo de tres vías y cambia al estado *ESTABLISHED*. Ahora se pueden enviar y recibir datos.

Cuando una aplicación ha terminado, ejecuta una primitiva CLOSE; esto hace que la entidad TCP local envíe un segmento *FIN* y espere el *ACK* correspondiente (el recuadro punteado con la leyenda “cierre activo”). Al llegar el *ACK*, se hace una transición al estado *FIN WAIT 2* y se cierra un sentido de la conexión. Cuando cierra también el otro lado llega un *FIN*, para el cual se envía una confirmación de recepción. Ahora ambos lados están cerrados, pero el TCP espera un tiempo igual al doble del tiempo de vida máximo del paquete para garantizar que todos los paquetes de la conexión hayan expirado, sólo por

si acaso se perdió la confirmación de recepción. Al expirar el temporizador, TCP borra el registro de la conexión.

Examinemos ahora la administración de la conexión desde el punto de vista del servidor. El servidor emite una solicitud LISTEN y se detiene a esperar a que aparezca alguien. Cuando llega un *SYN*, se envía una confirmación de recepción y el servidor pasa al estado *SYN RCVD*. Cuando llega la confirmación de recepción del *SYN* del servidor, el acuerdo de tres vías se completa y el servidor pasa al estado *ESTABLISHED*. Ahora puede ocurrir la transferencia de datos.

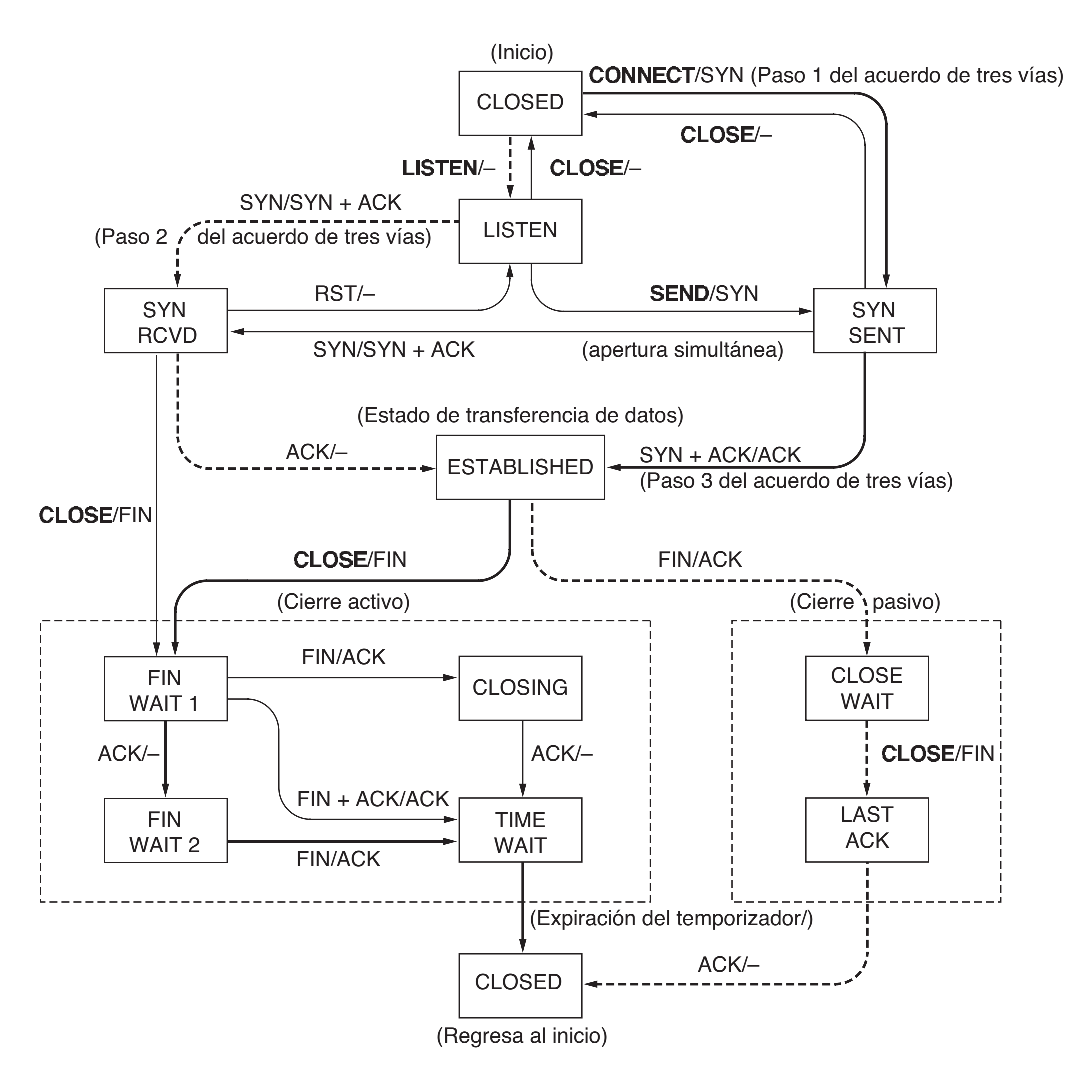


Figura 6-39. Máquina de estados finitos para administrar las conexiones TCP. La línea continua gruesa es la trayectoria normal para un cliente. La línea punteada gruesa es la trayectoria normal para un servidor. Las líneas delgadas son eventos no usuales. Cada transición se etiqueta con el evento que la ocasiona y la acción resultante, separada por una diagonal.

Cuando el cliente termina de transmitir sus datos, emite una solicitud CLOSE; esto provoca la llegada de un *FIN* al servidor (recuadro punteado con la leyenda "cierre pasivo"). Entonces se envía una señal al servidor. Cuando éste también emite una solicitud CLOSE, se envía un *FIN* al cliente. Al llegar la confirmación de recepción del cliente, el servidor libera la conexión y elimina el registro de conexión.

6.5.8 Ventana deslizante de TCP

Como ya vimos, la administración de ventanas en l TCP separa los aspectos de la confirmación de la recepción correcta de los segmentos y la asignación del búfer en el receptor. Por ejemplo, suponga que el receptor tiene un búfer de 4 096 bytes, como se muestra en la figura 6-40. Si el emisor transmite un segmento de 2 048 bytes que se recibe correctamente, el receptor enviará la confirmación de recepción del segmento. Sin embargo, dado que ahora sólo tiene 2 048 bytes de espacio de búfer (hasta que la aplicación retire algunos datos de éste), anunciará una ventana de 2048 comenzando con el siguiente byte esperado.

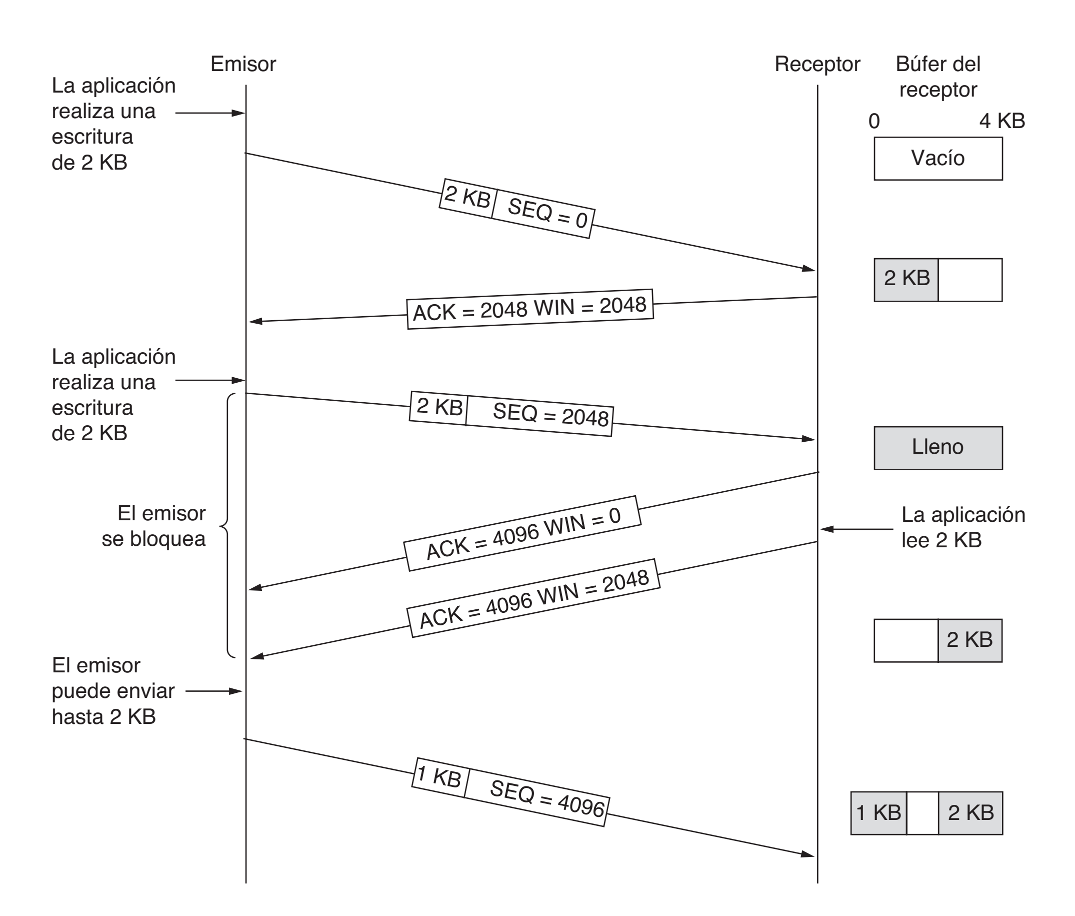


Figura 6-40. Administración de ventanas en TCP.

Ahora el emisor envía otros 2 048 bytes, para los cuales el receptor envía la confirmación de recepción, pero la ventana anunciada tiene un tamaño de 0. El emisor debe detenerse hasta que el proceso de aplicación en el host receptor retire algunos datos del búfer, momento en el que TCP podrá anunciar una ventana más grande y se podrán enviar más datos.

Cuando la ventana es de 0, el emisor por lo general no puede enviar segmentos, salvo en dos situaciones. En primer lugar, se pueden enviar datos urgentes (por ejemplo, para permitir que el usuario elimine el proceso en ejecución en la máquina remota). En segundo lugar, el emisor puede enviar un segmento de 1 byte para hacer que el receptor vuelva a anunciar el siguiente byte esperado y el tamaño de la ventana. Este paquete se denomina **sonda de ventana**. El estándar TCP proporciona explícitamente esta opción para evitar un interbloqueo si llega a perderse una actualización de ventana.

No se requiere que los emisores transmitan datos tan pronto como llegan de la aplicación. Tampoco se requiere que los receptores envíen confirmaciones de recepción tan pronto como sea posible. Por ejemplo, en la figura 6-40 cuando llegaron los primeros 2 KB de datos, TCP, a sabiendas que tenía disponible una ventana de 4 KB, hubiera actuado perfectamente bien si sólo almacenara en el búfer los datos hasta que llegaran otros 2 KB para transmitir un segmento con una carga útil de 4 KB. Podemos explotar esta libertad para mejorar el desempeño.

Considere una conexión a una terminal remota; por ejemplo, mediante el uso de SSH o telnet, que reacciona con cada pulso de tecla. En el peor de los casos, al llegar un carácter a la entidad TCP emisora, TCP crea un segmento TCP de 21 bytes que entrega al IP para que lo envíe como datagrama IP de 41 bytes. Del lado receptor, TCP envía de inmediato una confirmación de recepción de 40 bytes (20 bytes de encabezado TCP y 20 bytes de encabezado IP). Después, cuando la terminal remota lee el byte, TCP envía una actualización de ventana y recorre la ventana 1 byte hacia la derecha. Este paquete también es de 40 bytes. Por último, cuando la terminal remota procesa el carácter, lo retransmite para que se despliegue en forma local mediante un paquete de 41 bytes. En conjunto se usan 162 bytes de ancho de banda y se envían cuatro segmentos por cada carácter pulsado. Cuando el ancho de banda escasea, no es deseable este método de operación.

Un enfoque que usa muchas implementaciones de TCP para optimizar esta situación es el de las **confirmaciones de recepción con retardo**. La idea es retrasar las confirmaciones de recepción y las actualizaciones de ventana por hasta 500 mseg, con la esperanza de que lleguen algunos datos con los cuales se pueda viajar de manera gratuita. Suponiendo que la terminal hace eco en un lapso de 500 mseg, ahora el lado remoto sólo necesita enviar de vuelta un paquete de 41 bytes, con lo cual se recorta a la mitad la cuenta de paquetes y el uso de ancho de banda.

Aunque las confirmaciones de recepción con retardo reducen la carga impuesta en la red por el receptor, un emisor que envía varios paquetes cortos (por ejemplo, paquetes de 41 bytes que contengan 1 byte de datos) aún opera de manera ineficiente. El **algoritmo de Nagle** (Nagle, 1984) es una manera de reducir este uso. Lo que sugirió Nagle es sencillo: cuando llegan datos en pequeñas piezas al emisor, sólo se envía la primera pieza y el resto se almacena en búfer hasta que se confirma la recepción del byte pendiente. Después se envían todos los datos del búfer en un segmento TCP y nuevamente comienzan a almacenarse en búfer los datos hasta que se haya confirmado la recepción del siguiente segmento. Esto significa que sólo puede haber un paquete corto pendiente en cualquier momento dado. Si la aplicación envía muchas piezas de datos en el tiempo de ida y vuelta, el algoritmo de Nagle colocará todas las diversas piezas en un segmento, con lo cual se reducirá de manera considerable el ancho de banda utilizado. Además, el algoritmo establece que se debe enviar un nuevo segmento si se acumularon suficientes datos como para llenar un segmento máximo.

El algoritmo de Nagle se usa mucho en las implementaciones de TCP, pero hay veces en que es mejor deshabilitarlo. En particular, en los juegos interactivos que operan a través de Internet, por lo general los jugadores desean un flujo rápido de paquetes cortos de actualización. Si se acumulan las actualizaciones para enviarlas en ráfagas, el juego responderá de manera errática, lo cual provocará que los usuarios estén molestos. Un problema más sutil es que en ocasiones el algoritmo de Nagle puede interactuar con las confirmaciones de recepción con retardo para provocar un interbloqueo temporal: el receptor espera los datos sobre los cuales superponer una confirmación de recepción, y el emisor espera la confirmación de la recepción para enviar más datos. Esta interacción puede retardar las descargas de las páginas web. Debido a estos problemas, el algoritmo de Nagle se puede deshabilitar (lo cual se conoce como la opción *TCP_NODELAY*). Mogul y Minshall (2001) analizan ésta y otras soluciones.

Otro problema que puede arruinar el desempeño de TCP es el **síndrome de ventana tonta** (Clark, 1982). Este problema ocurre cuando se pasan datos a la entidad TCP emisora en bloques grandes, pero una aplicación interactiva del lado receptor lee datos sólo a razón de 1 byte a la vez. Para ver el problema, analice la figura 6-41. En un principio, el búfer TCP del lado receptor está lleno y el emisor lo sabe (es decir, tiene un

tamaño de ventana de 0). Entonces la aplicación interactiva lee un carácter del flujo TCP. Esta acción hace feliz al receptor TCP, por lo que envía una actualización de ventana al emisor para indicar que está bien que envíe 1 byte. El emisor accede y envía 1 byte. El búfer ahora está lleno, por lo que el receptor confirma la recepción del segmento de 1 byte y establece la ventana a 0. Este comportamiento puede continuar por siempre.

La solución de Clark es evitar que el receptor envíe una actualización de ventana para 1 byte. En cambio, se le obliga a esperar hasta tener disponible una cantidad decente de espacio, y luego lo anuncia. Específicamente, el receptor no debe enviar una actualización específica de ventana sino hasta que pueda manejar el tamaño máximo de segmento que anunció al establecerse la conexión, o hasta que su búfer quede a la mitad de capacidad, lo que sea más pequeño. Además, el emisor también puede ayudar al no enviar segmentos muy pequeños. En cambio, debe esperar hasta que pueda enviar un segmento completo, o por lo menos uno que contenga la mitad del tamaño del búfer del receptor.

El algoritmo de Nagle y la solución de Clark al síndrome de ventana tonta son complementarios. Nagle trataba de resolver el problema causado por la aplicación emisora que entregaba datos a TCP, 1 byte a la vez. Clark trataba de resolver el problema de que la aplicación receptora tomara los datos de TCP, 1 byte a la vez. Ambas soluciones son válidas y pueden operar juntas. El objetivo es que el emisor no envíe segmentos pequeños y que el receptor no los pida.

El receptor TCP también puede hacer más para mejorar el desempeño que sólo actualizar ventanas en unidades grandes. Al igual que el emisor TCP, tiene la capacidad de almacenar datos en el búfer, por lo que puede bloquear una solicitud READ de la aplicación hasta que pueda proporcionarle un bloque grande de datos. Al hacer esto se reduce la cantidad de llamadas a TCP (y la sobrecarga). También aumenta el tiempo de respuesta, pero en las aplicaciones no interactivas tales como la transferencia de archivos, la eficiencia puede ser más importante que el tiempo de respuesta a las solicitudes individuales.

Otro problema que el receptor debe solucionar es que los segmentos pueden llegar fuera de orden. El receptor colocará los datos en el búfer hasta que se puedan pasar en orden a la aplicación. En realidad no ocurriría nada malo si se descartaran los segmentos fuera de orden, ya que el emisor los retransmitiría en un momento dado, pero sería un desperdicio.

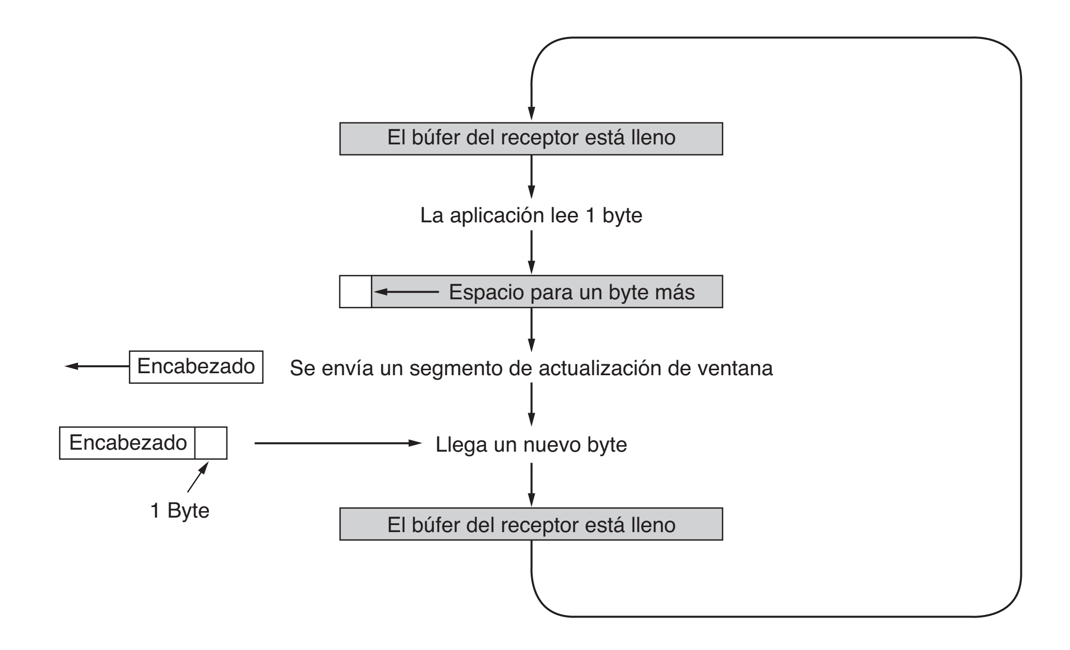


Figura 6-41. Síndrome de ventana tonta.

Las confirmaciones de recepción se pueden enviar sólo después de haber recibido todos los datos hasta el byte confirmado. A esto se le conoce como **confirmación de recepción acumulativa**. Si el receptor recibe los segmentos 0, 1, 2, 4, 5, 6 y 7, puede enviar una confirmación de recepción de todos los bytes hasta el último byte del segmento 2, inclusive. Al expirar el temporizador del emisor, éste retransmitirá el segmento 3. Como el receptor ha puesto en el búfer los segmentos 4 a 7, al recibir el segmento 3 puede enviar una confirmación de recepción de todos los bytes hasta el final del segmento 7.

6.5.9 Administración de temporizadores de TCP

TCP usa varios temporizadores (al menos de manera conceptual) para hacer su trabajo. El más importante de éstos es el **RTO** (**Temporizador de Retransmisión**, del inglés *Retransmission TimeOut*). Cuando se envía un segmento, se inicia un temporizador de retransmisiones. Si la confirmación de recepción del segmento llega antes de que expire el temporizador, éste se detiene. Por otro lado, si el temporizador termina antes de que llegue la confirmación de recepción, se retransmite el segmento (y se inicia de nuevo el temporizador). Surge entonces la pregunta: ¿qué tan grande debe ser el intervalo de expiración del temporizador?

Este problema es mucho más difícil en la capa de transporte que en los protocolos de enlace de datos tales como 802.11. En este último caso, el retardo esperado se mide en microsegundos y es muy predecible (es decir, tiene una varianza baja), por lo que el temporizador se puede establecer para expirar justo después del momento en que se esperaba la confirmación de recepción, como se muestra en la figura 6-42(a). Dado que las confirmaciones de recepción pocas veces se retardan en la capa de enlace de datos (debido a la falta de congestión), la ausencia de una confirmación de recepción en el momento esperado por lo general significa que se perdió la trama o la confirmación de recepción.

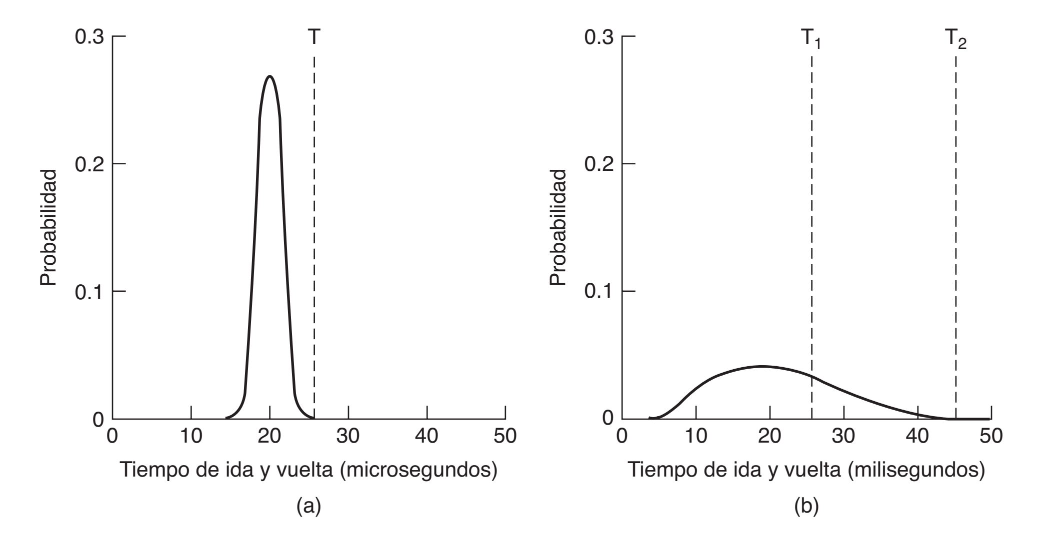


Figura 6-42. (a) Densidad de probabilidad de los tiempos de llegada de las confirmaciones de recepción en la capa de enlace de datos. (b) Densidad de probabilidad de los tiempos de llegada de las confirmaciones de recepción para TCP.

TCP se enfrenta a un entorno radicalmente distinto. La función de densidad de probabilidad del tiempo que tarda en regresar una confirmación de recepción TCP se parece más a la figura 6-42(b) que a la figura 6-42(a). Es más grande y variable. Es complicado determinar el tiempo de ida y vuelta al destino. Incluso cuando se conoce, es difícil decidir sobre el intervalo de expiración del temporizador. Si se establece demasiado corto, por decir T_1 en la figura 6-42(b), ocurrirán retransmisiones innecesarias e

Internet se llenará de paquetes inútiles. Si se establece demasiado largo (por ejemplo, T_2), el desempeño sufrirá debido al largo retardo de retransmisión de cada paquete perdido. Es más, la varianza y la media de la distribución de llegadas de confirmaciones de recepción pueden variar con rapidez en unos cuantos segundos, a medida que se generan y se resuelven congestionamientos.

La solución es usar un algoritmo dinámico que ajuste de manera constante el intervalo de expiración del temporizador, con base en mediciones continuas del desempeño de la red. El algoritmo utilizado por lo general por el TCP se lo debemos a Jacobson (1988) y funciona de la siguiente manera. Por cada conexión, TCP mantiene una variable llamada *SRTT* (Tiempo de Ida y Vuelta Suavizado), que es la mejor estimación actual del tiempo de ida y vuelta al destino en cuestión. Al enviarse un segmento, se inicia un temporizador, tanto para ver el tiempo que tarda la confirmación de recepción como para activar una retransmisión si se tarda demasiado. Si llega la confirmación de recepción antes de expirar el temporizador, TCP mide el tiempo que tardó la confirmación de recepción, por decir *R*. Luego actualiza el *SRTT* de acuerdo con la fórmula

$$SRTT = \alpha\, SRTT + (1 - \alpha)\, R$$

en donde α es un factor de suavizado que determina la rapidez con que se olvidan los valores anteriores. Por lo común, $\alpha = 7/8$. Este tipo de fórmula es un **EWMA** (**Promedio Móvil Ponderado Exponencialmente**, del inglés *Exponentially Weighted Moving Average*) o un filtro pasa bajas, que descarta el ruido en las muestras.

Incluso con un buen valor de *SRTT*, seleccionar la expiración adecuada del temporizador de retransmisión no es un asunto sencillo. Las primeras implementaciones de TCP usaban 2x*RTT*, pero la experiencia demostró que un valor constante era inflexible, puesto que no respondía cuando subía la varianza. En particular, los modelos de encolamiento de tráfico al azar (por ejemplo, Poisson) predicen que cuando la carga se acerca a la capacidad máxima, el retardo se hace grande y muy variable. Esto puede provocar que el temporizador de retransmisión se active y se retransmita una copia del paquete, aunque el paquete original aún esté transitando por la red. Es más probable que esto ocurra en condiciones de mucha carga, que viene siendo el peor momento para enviar paquetes adicionales por la red.

Para corregir este problema, Jacobson propuso hacer que el valor de expiración del temporizador fuera sensible a la diferencia en los tiempos de ida y vuelta, así como al tiempo de ida y vuelta suavizado. Para este cambio hay que llevar el registro de otra variable suavizada, *RTTVAR* (Variación de Tiempo de Ida y Vuelta) que se actualiza mediante la siguiente fórmula:

$$RTTVAR = \beta\, RTTVAR + (1 - \beta)\, |\, SRTT - R\, |$$

Ésta es también una fórmula EWMA; por lo general, $\beta = 3/4$. El tiempo de expiración de retransmisión, *RTO*, se establece así:

$$RTO = SRT + 4 \times RTTVAR$$

La elección del factor 4 es un tanto arbitraria, pero se puede hacer la multiplicación por 4 con un solo desplazamiento y menos de 1% de todos los paquetes llegan después de más de cuatro desviaciones estándar. Observe que *RTTVAR* no es exactamente lo mismo que la desviación estándar (en realidad es la desviación media), pero es lo bastante parecida en la práctica. La publicación de Jacobson está llena de trucos astutos para calcular los tiempos de expiración de los temporizadores con sólo usar sumas, restas y desplazamientos de enteros. Esta economía no es necesaria para los hosts modernos, pero se ha convertido en parte de la cultura que permite a TCP ejecutarse en todo tipo de dispositivos, desde supercomputadoras hasta pequeños dispositivos. Hasta ahora nadie ha logrado colocarlo dentro de un chip RFID, pero tal vez un día de éstos alguien lo logre.

En el RFC 2 988 se proporcionan más detalles acerca de cómo calcular este tiempo de expiración, incluyendo los valores iniciales de las variables. El temporizador de retransmisión también se mantiene en un mínimo de 1 segundo, sin importar las estimaciones. Éste es un valor conservador que se seleccionó para evitar retransmisiones espurias con base en las mediciones (Allman y Paxson, 1999).

Un problema que ocurre con la recopilación de las muestras, *R,* del tiempo de ida y vuelta es qué hacer cuando expira el temporizador de un segmento y se envía de nuevo. Cuando llega la confirmación de recepción, no está claro si ésta se refiere a la primera transmisión o a una posterior. Si adivinamos mal se puede contaminar seriamente el temporizador de retransmisión. Phil Karn descubrió este problema de la manera difícil. Él es un radioaficionado interesado en la transmisión de paquetes TCP/IP a través de la radio amateur, un medio notoriamente poco confiable. Karn hizo una propuesta sencilla: no actualizar las estimaciones sobre ninguno de los segmentos retransmitidos. Además, se duplicará el tiempo de expiración con cada retransmisión sucesiva hasta que los segmentos pasen a la primera. Esta corrección se conoce como **algoritmo de Karn** (Karn y Partridge, 1987). La mayoría de las implementaciones de TCP lo utilizan.

El temporizador de retransmisiones no es el único temporizador que TCP utiliza. El **temporizador de persistencia** es el segundo de ellos. Está diseñado para evitar el siguiente interbloqueo. El receptor envía una confirmación de recepción con un tamaño de ventana de 0 para indicar al emisor que espere. Después el receptor actualiza la ventana, pero se pierde el paquete con la actualización. Ahora, tanto el emisor como el receptor están esperando a que el otro haga algo. Cuando expira el temporizador de persistencia, el emisor transmite un sondeo al receptor. La respuesta al sondeo proporciona el tamaño de la ventana. Si aún es cero, se inicia el temporizador de persistencia una vez más y se repite el ciclo. Si es diferente de cero, ahora se pueden enviar datos.

Un tercer temporizador que utilizan algunas implementaciones es el **temporizador de seguir con vida** (*keepalive*). Cuando una conexión ha estado inactiva durante demasiado tiempo, el temporizador de seguir con vida puede expirar para ocasionar que un lado compruebe que el otro aún está ahí. Si no se recibe respuesta, se termina la conexión. Esta característica es controversial puesto que agrega sobrecarga y puede terminar una conexión saludable debido a una partición temporal de la red.

El último temporizador que se utiliza en cada conexión TCP es el que se usa en el estado TIME WAIT durante el cierre. Opera durante el doble del tiempo máximo de vida de paquete para asegurar que, al cerrarse una conexión, todos los paquetes creados por ella hayan desaparecido.

6.5.10 Control de congestión en TCP

Guardamos una de las funciones clave de TCP para lo último: el control de congestión. Cuando la carga ofrecida a cualquier red es mayor que la que puede manejar, se genera una congestión. Internet no es ninguna excepción. La capa de red detecta la congestión cuando las colas crecen demasiado en los enrutadores y trata de lidiar con este problema, aunque lo único que haga sea descartar paquetes. Es responsabilidad de la capa de transporte recibir la retroalimentación de congestión de la capa de red y reducir la tasa del tráfico que envía a la red. En Internet, TCP desempeña el papel principal en cuanto al control de la congestión, así como en el transporte confiable. Por esto se le considera un protocolo tan especial.

En la sección 6.3 vimos la situación general sobre el control de la congestión. Una conclusión clave fue la siguiente: un protocolo de transporte, que utiliza una ley de control AIMD (Incremento Aditivo/Decremento Multiplicativo) en respuesta a las señales de congestión binarias provenientes de la red, converge hacia una asignación de ancho de banda equitativa y eficiente. El control de congestión de TCP se basa en la implementación de esta metodología mediante el uso de una ventana y con la pérdida de paquetes como la señal binaria. Para ello, TCP mantiene una **ventana de congestión** cuyo tamaño es el número

de bytes que puede tener el emisor en la red en cualquier momento dado. La tasa correspondiente es el tamaño de ventana dividido entre el tiempo de viaje de ida y vuelta de la conexión. TCP ajusta el tamaño de la ventana, de acuerdo con la regla AIMD.

Recuerde que la ventana de congestión se mantiene *además* de la ventana de control de flujo, la cual especifica el número de bytes que el receptor puede colocar en el búfer. Ambas ventanas se rastrean en paralelo, y el número de bytes que se puede enviar es el número que sea menor de las dos ventanas. Por esto, la ventana efectiva es la cantidad más pequeña de lo que tanto el emisor como el receptor consideran que está bien. Se necesitan dos para bailar. TCP dejará de enviar datos si la ventana de congestión o la de control de flujo está temporalmente llena. Si el receptor dice "envía 64 KB" pero el emisor sabe que las ráfagas mayores de 32 KB obstruyen la red, enviará 32 KB. Por otro lado, si el receptor dice "envía 64 KB" y el emisor sabe que las ráfagas de hasta 128 KB pasan sin problema, enviará los 64 KB solicitados. Ya describimos antes la ventana de control de flujo, por lo que a continuación sólo describiremos la ventana de congestión.

El control de congestión moderno se agregó a TCP en gran parte gracias a los esfuerzos de Van Jacobson (1988). Es una historia fascinante. A partir de 1986, la creciente popularidad de Internet condujo a la primera ocurrencia de lo que más tarde se conoció como **colapso por congestión**, un periodo prolongado en donde el caudal útil se reducía en forma precipitada (es decir, por más de un factor de 100) debido a la congestión en la red. Jacobson (y muchos otros) buscaba comprender qué estaba ocurriendo para remediar la situación.

La corrección de alto nivel que implementó Jacobson era aproximar una ventana de congestión AIMD. La parte interesante, y una parte importante de la complejidad del control de congestión en TCP, es ver cómo Jacobson agregó esto a una implementación existente sin cambiar ninguno de los formatos de los mensajes, gracias a lo cual se podía implementar al instante. Para empezar, observó que la pérdida de paquetes es una señal adecuada de congestión. Esta señal llega un poco tarde (puesto que la red ya se encuentra congestionada) pero es bastante confiable. Después de todo, es difícil construir un enrutador que no descarte paquetes cuando está sobrecargado. No es muy probable que este hecho vaya a cambiar. Incluso cuando aparezcan memorias con capacidades de terabytes para colocar grandes cantidades de paquetes en el búfer, probablemente habrá redes de terabits/seg para llenar estas memorias.

Sin embargo, para usar la pérdida de paquetes como una señal de congestión es necesario que los errores de transmisión sean relativamente raros. Por lo general esto no es así para los enlaces inalámbricos como las redes 802.11, lo cual explica por qué incluyen su propio mecanismo de retransmisión en la capa de enlace. Debido a las retransmisiones inalámbricas, es común que la pérdida de paquetes en la capa de red debido a los errores de transmisión, se enmascare en las redes inalámbricas. También es algo raro en otros enlaces, ya que los cables y la fibra óptica por lo general tienen tasas bajas de error de bits.

Todos los algoritmos de TCP en Internet suponen que los paquetes perdidos se deben a la congestión, por lo cual monitorean las expiraciones de los temporizadores y buscan señales de problemas. Se requiere un buen temporizador de retransmisión para detectar las señales de pérdida de paquetes con precisión y en forma oportuna. Ya hemos visto cómo el temporizador de retransmisión de TCP incluye estimaciones de la media y la variación en los tiempos de ida y vuelta. Corregir este temporizador mediante la inclusión del factor de variación fue un paso importante en el trabajo realizado por Jacobson. Si se tiene un buen temporizador de retransmisión, el emisor de TCP puede rastrear el número pendiente de bytes que están cargando la red. Simplemente analiza la diferencia entre los números de secuencia que se transmiten y los números de secuencia cuya confirmación de recepción se ha enviado.

Ahora, tal parece que nuestra tarea es sencilla. Todo lo que tenemos que hacer es rastrear la ventana de congestión mediante el uso de los números de secuencia y los que ya se han confirmado, y ajustar la ventana de congestión mediante el uso de una regla AIMD. Pero como podríamos esperar, es más complicado que eso. Una de las primeras consideraciones es que la forma en que se envían los paquetes a la red, incluso a través de periodos cortos, debe coincidir con la trayectoria de red. En caso contrario, el

tráfico provocará una congestión. Por ejemplo, considere un host con una ventana de congestión de 64 KB conectado a una red Ethernet conmutada de 1 Gbps. Si el host envía toda la ventana a la vez, esta ráfaga de tráfico puede viajar a través de una línea ADSL lenta de 1 Mbps en un punto posterior en la trayectoria. La ráfaga que tardó sólo medio milisegundo en la línea de 1 Gbps obstruirá la línea de 1 Mbps durante medio segundo, lo cual perturbará por completo a los protocolos tales como el de voz sobre IP. Este comportamiento podría ser conveniente en un protocolo diseñado para provocar congestión, pero no en uno para controlarla.

Sin embargo, resulta ser que podemos usar pequeñas ráfagas de paquetes para nuestra ventaja. La figura 6-43 muestra qué ocurre cuando un emisor en una red rápida (el enlace de 1 Gbps) envía una pequeña ráfaga de cuatro paquetes a un receptor en una red lenta (el enlace de 1 Mbps), que constituye el cuello de botella o la parte más lenta de la trayectoria. En un principio los cuatro paquetes viajan a través del enlace lo más rápido que el emisor puede enviarlos. En el enrutador se ponen en cola mientras se envían, ya que es más tardado enviar un paquete a través del enlace lento que recibir el siguiente paquete a través del enlace rápido. Pero la cola no es grande debido a que sólo se envió un pequeño número de paquetes a la vez. Observe el aumento en la longitud de los paquetes en el enlace lento. El mismo paquete, por decir de 1 KB, ahora es más largo debido a que se requiere más tiempo para enviarlo por un enlace lento que por uno rápido.

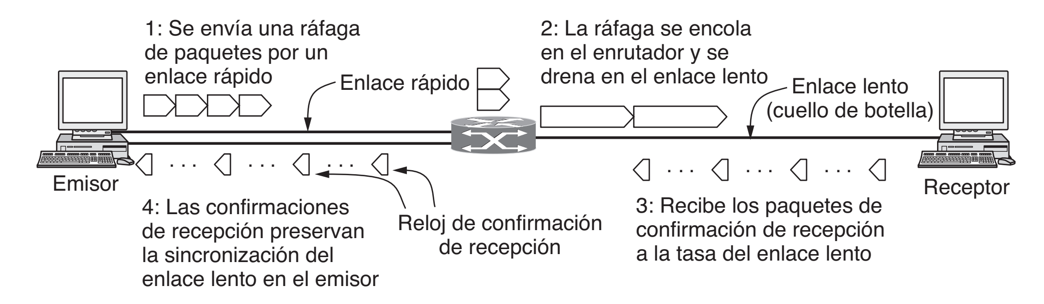


Figura 6-43. Una ráfaga de paquetes de un emisor y el reloj de confirmación de recepción devuelto.

En un momento dado, los paquetes llegan al receptor y se confirma su recepción. Los tiempos para las confirmaciones de recepción reflejan los tiempos en los que llegaron los paquetes al receptor después de atravesar el enlace lento. Están dispersos en comparación con los paquetes originales en el enlace rápido. Mientras estas confirmaciones de recepción viajan a través de la red y regresan al emisor, preservan esta sincronización.

La observación clave es ésta: las confirmaciones de recepción regresan al emisor aproximadamente a la misma tasa a la que se pueden enviar los paquetes a través del enlace más lento en la trayectoria. Ésta es la tasa que el emisor quiere usar. Si inyecta nuevos paquetes en la red a esta tasa, se enviarán tan rápido como lo permita el enlace lento, pero no se encolarán ni congestionarán ningún enrutador a lo largo de la trayectoria. Esta sincronización se conoce como **reloj de confirmación de recepción** (*ack clock*) y es una parte esencial de TCP. Mediante el uso de un reloj de confirmación de recepción, TCP regula el tráfico y evita las colas innecesarias en los enrutadores.

Una segunda consideración es que la regla AIMD tardará mucho tiempo para llegar a un buen punto de operación en las redes rápidas si la ventana de congestión se inicia a partir de un tamaño pequeño. Considere una trayectoria de red modesta que puede soportar 10 Mbps con un RTT de 100 milisegundos. La ventana de congestión apropiada es el producto ancho de banda-retardo, que es de 1 Mbit, o 100 paquetes de 1250 bytes cada uno. Si la ventana de congestión empieza en 1 paquete y se incrementa 1 paquete por cada RTT, transcurrirán 100 RTT o 10 segundos antes de que la conexión opere a la tasa correcta aproximada. Hay que esperar mucho tiempo sólo para llegar a la velocidad correcta para una transferencia.

Podríamos reducir este tiempo de inicio si empezáramos con una ventana inicial más grande, por decir de 50 paquetes. Pero esta ventana sería demasiado grande para los enlaces lentos o cortos. Provocaría congestión si se utilizara toda a la vez, como hemos descrito antes.

En cambio, la solución que Jacobson eligió para manejar ambas consideraciones es una mezcla de incremento lineal y de incremento multiplicativo. Al establecer una conexión, el emisor inicializa la ventana de congestión con un pequeño valor inicial, de cuando menos cuatro segmentos; los detalles se describen en el RFC 3 390 y el uso de cuatro segmentos es una mejoría respecto a un valor inicial anterior de un segmento, con base en la experiencia. A continuación el emisor envía la ventana inicial. Los paquetes tardarán un tiempo de ida y vuelta para que se confirme su recepción. Para cada segmento cuya recepción se confirme antes de que expire el temporizador de retransmisión, el emisor agrega a la ventana de congestión una cantidad de bytes equivalente a un segmento. Además, como ya se confirmó la recepción de ese segmento, ahora hay un segmento menos en la red. El resultado es que cada segmento confirmado permite enviar dos segmentos más. La ventana de congestión se duplica cada tiempo de ida y vuelta.

Este algoritmo se conoce como **inicio lento** (*slow start*), pero no es para nada lento (su crecimiento es exponencial), excepto en comparación con el algoritmo anterior que permitía enviar toda una ventana de control de flujo al mismo tiempo. El inicio lento se muestra en la figura 6-44. En el primer tiempo de ida y vuelta, el emisor inyecta un paquete a la red (y el receptor recibe un paquete). En el siguiente tiempo de ida y vuelta se envían dos paquetes, y en el tercer tiempo de ida y vuelta cuatro paquetes.

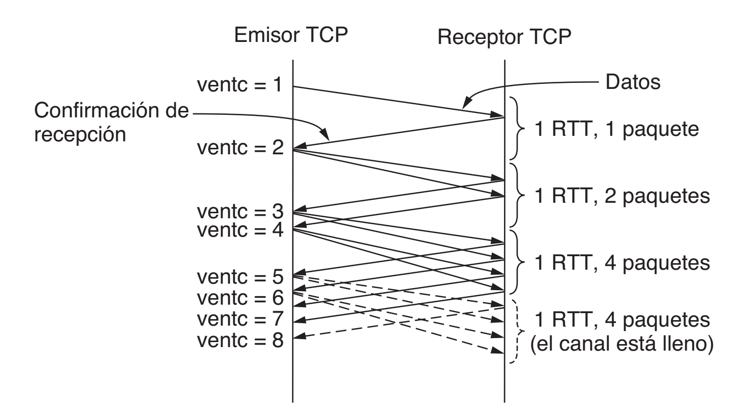


Figura 6-44. Inicio lento desde una ventana de congestión inicial de un segmento.

El inicio lento funciona bien sobre un rango de velocidades de enlaces y tiempos de ida y vuelta; además utiliza un reloj de confirmación de recepción para asociar la tasa de transmisiones del emisor con la trayectoria de red. En la figura 6-44 puede observar la forma en que las confirmaciones de recepción regresan del emisor al receptor. Cuando el emisor recibe una confirmación de recepción, incrementa en uno la ventana de congestión y envía de inmediato dos paquetes a la red (un paquete es el incremento en uno; el otro paquete es un reemplazo para el paquete cuya confirmación se recibió y salió de la red. La ventana de congestión proporciona en todo momento el número de paquetes sin confirmación de recepción). Sin embargo, estos dos paquetes no llegarán necesariamente al receptor con un espaciamiento tan estrecho como cuando se enviaron. Por ejemplo, suponga que el emisor se encuentra en una red Ethernet de 100 Mbps. Cada paquete de 1250 bytes tarda 100 μseg en enviarse. Así, el retardo entre los paquetes puede ser como mínimo de 100 μseg. La situación cambia si estos paquetes pasan a través de un enlace ADSL de 1 Mbps en cualquier parte a lo largo de la trayectoria. Ahora se requieren 10 mseg para enviar el mismo paquete. Esto significa que el espaciamiento mínimo entre los dos paquetes aumentó por un factor de 100.

A menos que los paquetes tengan que esperar juntos en una cola de un enlace posterior, el espaciamiento seguirá siendo grande.

En la figura 6-44 se muestra este efecto al imponer un espaciamiento mínimo entre los paquetes de datos que llegan al receptor. Se mantiene el mismo espaciamiento cuando el receptor envía confirmaciones de recepción, y también cuando el emisor recibe esas confirmaciones. Si la trayectoria de la red es lenta, las confirmaciones de recepción llegarán con lentitud (después de un retardo de un RTT). Si la trayectoria de la red es rápida, las confirmaciones de recepción llegarán con rapidez (de nuevo, después del RTT). Todo lo que tiene que hacer el emisor es seguir la sincronización del reloj de confirmación de recepción mientras inyecta nuevos paquetes; esto es lo que hace el inicio lento.

Como el inicio lento provoca un crecimiento exponencial, en un momento dado (y más pronto que tarde) enviará demasiados paquetes a la red con mucha rapidez. Cuando esto ocurra, las colas se acumularán en la red. Cuando las colas estén llenas se perderán uno o más paquetes. Una vez que ocurra esto, el temporizador del emisor TCP expirará cuando una confirmación de recepción no llegue a tiempo. Hay evidencia de que el inicio lento aumenta con demasiada rapidez en la figura 6-44. Después de tres RTT, hay cuatro paquetes en la red. Estos cuatro paquetes tardan todo un RTT en llegar al receptor. Es decir, una ventana de congestión de cuatro paquetes es el tamaño correcto para esta conexión. Sin embargo, a medida que se confirma la recepción de estos paquetes, el inicio lento continúa aumentando el tamaño de la ventana de congestión y llega a ocho paquetes en otro RTT. Sólo cuatro de estos paquetes pueden llegar al receptor en un RTT, sin importar cuántos se envíen. Es decir, el canal de la red está lleno. Los paquetes adicionales que el emisor coloque en la red se acumularán en las colas de los enrutadores, ya que no pueden entregarse al receptor con la suficiente rapidez. Pronto ocurrirá una congestión y se perderán paquetes.

Para mantener el inicio lento bajo control, el emisor mantiene un umbral para la conexión, conocido como **umbral de inicio lento**. En un principio este valor se establece en un nivel arbitrariamente alto, al tamaño de la ventana de control de flujo, de manera que no limite la conexión. TCP continúa incrementando la ventana de congestión en el inicio lento hasta que expira un temporizador o la ventana de congestión excede el umbral (o cuando se llena la ventana del receptor).

Por ejemplo, cada vez que se detecta la pérdida de un paquete debido a la expiración de un temporizador, el umbral de inicio lento se establece a la mitad de la ventana de congestión y se reinicia todo el proceso. La idea es que la ventana actual es demasiado grande, puesto que antes provocó una congestión que ahora sólo se puede detectar mediante la expiración de un temporizador. Quizá la mitad de la ventana, que se utilizó con éxito en un tiempo anterior, sea una mejor estimación para una ventana de congestión que está cerca de la capacidad de la trayectoria y no provocará pérdida. En nuestro ejemplo de la figura 6-44, si se aumenta el tamaño de la ventana de congestión a ocho paquetes se puede provocar una pérdida, mientras que la ventana de congestión de cuatro paquetes en el RTT anterior tenía el valor correcto. Así, la ventana de congestión se restablece a su valor pequeño inicial y se reanuda el inicio lento.

Cada vez que se atraviesa el umbral de inicio lento, TCP cambia del inicio lento al incremento aditivo. En este modo, la ventana de congestión se incrementa un segmento por cada tiempo de ida y vuelta. Al igual que el inicio lento, por lo general esto se implementa mediante un incremento por cada segmento cuya recepción se ha confirmado, en vez de usar un incremento una vez por cada RTT. Llamemos *ventc* a la ventana de congestión y *MSS* al tamaño de segmento máximo. Una aproximación común sería incrementar *ventc* mediante la fórmula $(MSS \times MSS)/ventc$ por cada uno de los $ventc/MSS$ paquetes cuya recepción se pueda confirmar. Este incremento no necesita ser rápido. La idea general es que una conexión TCP pase mucho tiempo con su ventana de congestión cerca del valor óptimo: no tan pequeño como para que la tasa de transferencia real sea baja y no tan grande como para que ocurra una congestión.

El incremento aditivo se muestra en la figura 6-45 para la misma situación que el inicio lento. Al final de cada RTT, la ventana de congestión del emisor ha crecido lo suficiente como para inyectar un paquete

adicional a la red. En comparación con el inicio lento, la tasa lineal de crecimiento es mucho menor. No hay mucha diferencia para las ventanas de congestión pequeñas, como en este caso, sino una gran diferencia en el tiempo que se requiere para hacer crecer la ventana de congestión hasta 100 segmentos, por ejemplo.

También hay algo más que podemos hacer para mejorar el desempeño. El defecto en el esquema hasta ahora es esperar a que expire un temporizador. Los tiempos de expiración son algo largos debido a que deben ser conservadores. Después de perder un paquete, el receptor no puede confirmar la recepción más allá de ese paquete, por lo que el número de confirmación de recepción permanecerá fijo y el emisor no podrá enviar nuevos paquetes a la red, debido a que su ventana de congestión permanecerá llena. Esta condición puede continuar durante un periodo relativamente largo, hasta que el temporizador se dispare y se retransmita el paquete perdido. En ese punto, TCP vuelve a empezar con el inicio lento.

Hay una forma rápida de que el emisor reconozca que se ha perdido uno de sus paquetes. A medida que llegan al receptor los paquetes subsiguientes al perdido, activan confirmaciones de recepción que regresan al emisor. Estas confirmaciones de recepción contienen el mismo número de confirmación de recepción y, por ende, se denominan **confirmaciones de recepción duplicadas**. Cada vez que el emisor recibe una confirmación de recepción duplicada, es probable que haya llegado otro paquete al receptor y que el paquete perdido todavía no aparezca.

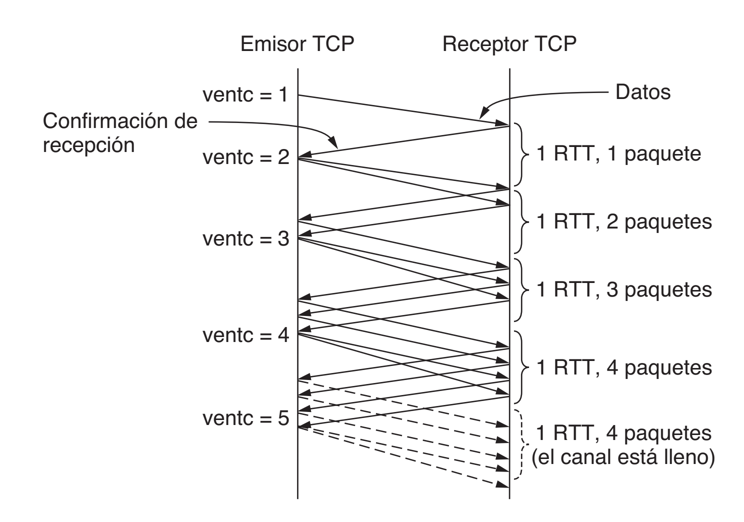


Figura 6-45. Incremento aditivo de una ventana de congestión inicial de un segmento.

Como los paquetes pueden tomar distintas trayectorias por la red, pueden llegar fuera de orden. Esto activará confirmaciones de recepción duplicadas, incluso aunque no se hayan perdido paquetes. Sin embargo, la mayor parte del tiempo esto es poco común en Internet. Cuando hay un reordenamiento a través de múltiples trayectorias, por lo general los paquetes recibidos no se reordenan demasiado. Así, TCP asume con cierta arbitrariedad que tres confirmaciones de recepción duplicadas indican que se ha perdido un paquete. La identidad del paquete perdido se puede inferir también del número de confirmación de recepción. Es el siguiente paquete inmediato en la secuencia. Entonces este paquete se puede retransmitir de inmediato, antes de que se dispare el temporizador de retransmisión.

Esta heurística se denomina **retransmisión rápida**. Una vez que se dispara, el umbral de inicio lento sigue establecido a la mitad de la ventana de congestión actual, justo como cuando expira un temporizador. Para volver a comenzar con el inicio lento, hay que establecer la ventana de congestión a un paquete. Con este tamaño de ventana, se enviará un nuevo paquete después del tiempo de ida y vuelta que se

requiere para confirmar la recepción del paquete retransmitido, junto con los datos que se habían enviado antes de detectar la pérdida.

En la figura 6-46 se muestra una ilustración del algoritmo de congestión que hemos construido hasta ahora. Esta versión de TCP se conoce como TCP Tahoe, en honor de la versión 4.2BSD Tahoe de 1988, en la que se incluyó. Aquí, el tamaño máximo de segmento es de 1 KB. En un principio la ventana de congestión era de 64 KB, pero ocurrió una expiración de temporizador, por lo que el umbral se estableció a 32 KB y la ventana de congestión a 1 KB para la transmisión 0. La ventana de congestión crece en forma exponencial hasta que llega al umbral (32 KB). La ventana se incrementa cada vez que llega una nueva confirmación de recepción, en vez de hacerlo en forma continua, lo cual provoca el patrón discreto de escalera. Una vez que se traspasa el umbral, la ventana crece en forma lineal. Se incrementa en un segmento por cada RTT.

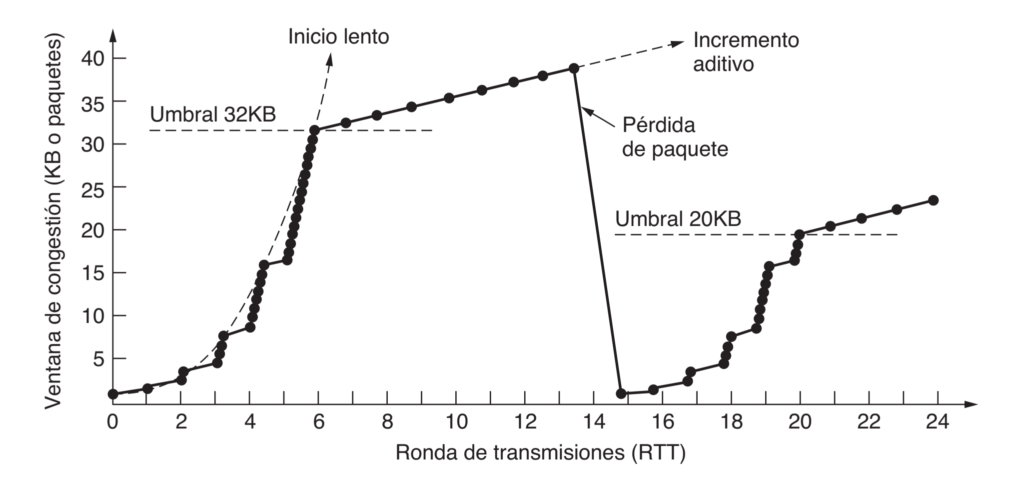


Figura 6-46. Inicio lento seguido de un incremento aditivo en el TCP Tahoe.

Las transmisiones en la ronda 13 son desafortunadas (deberían haberlo sabido), por lo que se pierde una de ellas en la red. Esto se detecta cuando llegan tres confirmaciones de recepción duplicadas. En ese momento se retransmite el paquete perdido, el umbral se establece a la mitad de la ventana actual (que para ahora es de 40 KB, por lo que la mitad es 20 KB) y se empieza de nuevo el proceso de inicio lento. Al reiniciar con una ventana de congestión de un paquete, se requiere un tiempo de ida y vuelta para que todos los datos transmitidos con anterioridad salgan de la red y se confirme su recepción, incluyendo el paquete retransmitido. La ventana de congestión crece con el inicio lento, como lo hizo antes, hasta que llega al nuevo umbral de 20 KB. En ese momento el crecimiento se vuelve lineal otra vez. Continuará de esta forma hasta que se detecte la pérdida de otro paquete mediante confirmaciones de recepción duplicadas o al expirar un temporizador (o cuando la ventana del receptor se convierta en el límite).

El TCP Tahoe (que incluía buenos temporizadores de retransmisión) ofrecía un algoritmo de control de congestión funcional que resolvió el problema del colapso por congestión. Jacobson descubrió que es posible hacerlo aún mejor. Al momento de la retransmisión rápida, la conexión está operando con una ventana de congestión demasiado grande, pero aún sigue operando con un reloj de confirmación de recepción funcional. Cada vez que llega otra confirmación de recepción duplicada, es probable que otro paquete haya dejado la red. Al usar las confirmaciones de recepción duplicadas para contabilizar los paquetes en la red, es posible dejar que algunos paquetes salgan de la red y continúen enviando uno nuevo para cada confirmación de recepción duplicada adicional.

La **recuperación rápida** es la heurística que implementa este comportamiento. Es un modo temporal que busca mantener el reloj de confirmación de recepción en operación con una ventana de congestión que es el nuevo umbral, o la mitad del valor de la ventana de congestión al momento de la retransmisión rápida. Para ello, se contabilizan las confirmaciones de recepción duplicadas (incluyendo las tres que desencadenaron la retransmisión rápida) hasta que el número de paquetes en la red disminuye a un valor equivalente al nuevo umbral. Esto requiere alrededor de medio tiempo de ida y vuelta. De ahí en adelante, se puede enviar un nuevo paquete por cada confirmación de recepción duplicada que se reciba. Un tiempo de ida y vuelta después de la retransmisión rápida se habrá confirmado la recepción del paquete perdido. En ese momento cesará el flujo de confirmaciones de recepción duplicadas y terminará el modo de recuperación rápida. Ahora la ventana de congestión se establecerá al nuevo umbral de inicio lento y crecerá mediante un incremento lineal.

El resultado de esta heurística es que TCP evita el inicio lento, excepto cuando la conexión se inicia por primera vez y cuando expira un temporizador. Esto último puede aún ocurrir cuando se pierde más de un paquete y la retransmisión rápida no recupera en forma adecuada. En vez de inicios lentos repetidos, la ventana de congestión de una conexión funcional sigue un patrón de **diente de sierra** de incremento aditivo (un segmento por cada RTT) y decremento multiplicativo (la mitad en un RTT). Ésta es exactamente la regla AIMD que buscábamos implementar.

En la figura 6-47 se muestra este comportamiento de diente de sierra. Se produce mediante el TCP Reno, nombrado en honor a la versión 4.3BSD Reno de 1990, en la cual se incluyó. En esencia, el TCP Reno es el TCP Tahoe más la recuperación rápida. Después de un inicio lento al principio, la ventana de congestión sube en forma lineal hasta que se detecta la pérdida de un paquete mediante confirmaciones de recepción duplicadas. El paquete perdido se retransmite y se utiliza la recuperación rápida para mantener el reloj de confirmación de recepción funcionando hasta que se confirma la recepción de la retransmisión. En ese momento, la ventana de congestión se reanuda a partir del nuevo umbral de inicio lento, en vez de empezar desde 1. Este comportamiento continúa en forma indefinida, y la conexión pasa la mayor parte del tiempo con su ventana de congestión cerca del valor óptimo del producto ancho de banda-retardo.

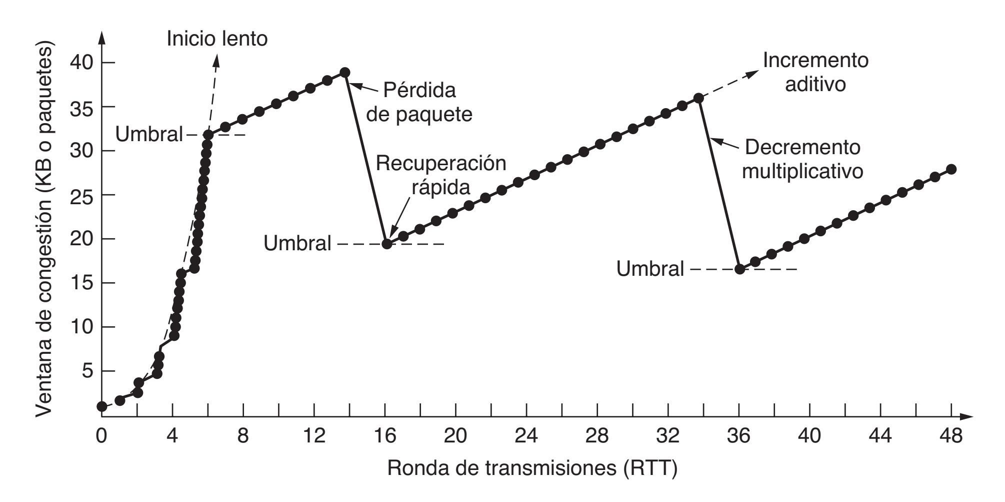


Figura 6-47. La recuperación rápida y el patrón de diente de sierra de TCP Reno.

Con sus mecanismos para ajustar la ventana de congestión, el TCP Reno ha formado la base para el control de congestión en TCP durante más de dos décadas. La mayoría de los cambios en los años intermedios han ajustado estos mecanismos en pequeñas formas; por ejemplo, al cambiar las opciones de la

ventana inicial y eliminar varias ambigüedades. Se han logrado algunas mejoras para recuperarse de dos o más pérdidas en una ventana de paquetes. Por ejemplo, la versión TCP NewReno usa un avance parcial del número de confirmación de recepción después de una retransmisión para encontrar y reparar otra pérdida (Hoe, 1996), según lo descrito en el RFC 3782. A mediados de la década de 1990 surgieron diversas variaciones que siguen los principios que hemos descrito, pero usan leyes de control un poco distintas. Por ejemplo, Linux usa una variante llamada CUBIC TCP (Ha y colaboradores, 2008), y Windows incluye una variante llamada Compound TCP (Tan y colaboradores, 2006).

También hay dos cambios más grandes que han afectado las implementaciones de TCP. En primer lugar, gran parte de la complejidad de TCP proviene de inferir mediante un flujo de confirmaciones de recepción duplicadas qué paquetes han llegado y cuáles se han perdido. El número de confirmación de recepción acumulativa no provee esta información. Una corrección simple es el uso de **SACK** (**Confirmaciones de Recepción Selectivas**, del inglés *Selective ACKnowledgements*), que lista hasta tres rangos de bytes que se hayan recibido. Con esta información, el emisor puede decidir de una manera más directa qué paquetes retransmitir, para así rastrear los paquetes en tránsito e implementar la ventana de congestión.

Cuando el emisor y el receptor establecen una conexión, cada uno envía la opción *SACK permitido* de TCP para indicar que comprenden las confirmaciones de recepción selectivas. Una vez que se habilita SACK para una conexión, funciona como se muestra en la figura 6-48. Un receptor usa el campo *Número de confirmación de recepción* de TCP de la manera usual, como una confirmación de recepción acumulativa del byte en orden más alto que se haya recibido. Cuando recibe el paquete 3 fuera de orden (debido a que se perdió el paquete 2), envía una *opción SACK* para los datos recibidos, junto con la confirmación de recepción acumulativa (duplicada) para el paquete 1. La *opción SACK* proporciona los rangos de bytes que se han recibido por encima del número proporcionado por la confirmación de recepción acumulativa. El primer rango es el paquete que desencadenó la confirmación de recepción duplicada. Los siguientes rangos, si están presentes, son bloques anteriores. Por lo común se utilizan hasta tres rangos. Para cuando se recibe el paquete 6, se utilizan dos rangos de bytes SACK para indicar que se recibieron el paquete 6 y los paquetes 3 a 4, además de todos los paquetes hasta el 1. De la información en cada *opción SACK* que recibe, el emisor puede decidir qué paquetes retransmitir. En este caso, sería una buena idea retransmitir los paquetes 2 y 5.

SACK es información estrictamente de asesoría. La detección real de la pérdida mediante el uso de confirmaciones de recepción duplicadas y ajustes a la ventana de congestión se lleva a cabo de la misma manera que antes. Sin embargo, mediante SACK, TCP se puede recuperar con más facilidad de las situaciones en las que se pierden varios paquetes casi al mismo tiempo, ya que el emisor TCP sabe qué paquetes no se han recibido. Ahora SACK se implementa ampliamente. Se describe en el RFC 2883, y el control de congestión en TCP mediante el uso de SACK se describe en el RFC 3517.

El segundo cambio es el uso de **ECN** (**Notificación Explícita de Congestión**, del inglés *Explicit Congestion Notification*) además de la pérdida de paquetes como señal de congestión. ECN es un mecanismo de la capa de IP para notificar a los hosts sobre la congestión, el cual describimos en la sección 5.3.4. Con este mecanismo, el receptor TCP puede recibir señales de congestión de IP.

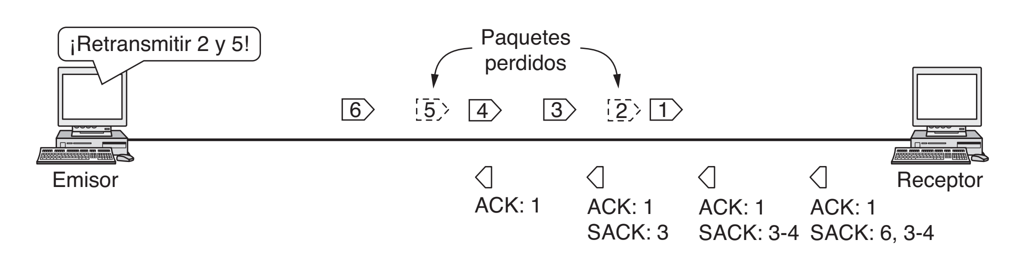


Figura 6-48. Confirmaciones de recepción selectivas.

El uso de ECN se habilita para una conexión TCP cuando tanto el emisor como el receptor indican que son capaces de usar ECN y activan los bits *ECE* y *CWR* al momento de establecer la conexión. Si se utiliza ECN, cada paquete que transporte un segmento TCP se señala con banderas en el encabezado IP para mostrar que puede transportar una señal ECN. Los enrutadores que soporten ECN establecerán una señal de congestión en los paquetes que puedan transportar banderas ECN cuando se aproxime la congestión, en vez de descartar esos paquetes después de que ocurra la congestión.

Se informa al receptor TCP si cualquier paquete que llegue transporta una señal de congestión de ECN. Luego el receptor utiliza la bandera *ECE* (*ECN Echo*) para indicar al emisor TCP que sus paquetes han experimentado una congestión. Para indicar al receptor que escuchó la señal, el emisor usa la bandera *CWR* (Ventana de Congestión Reducida).

El emisor TCP reacciona a estas notificaciones de congestión exactamente de la misma forma que reacciona con la pérdida de paquetes que se detecta mediante confirmaciones de recepción duplicadas. Sin embargo, la situación es mucho mejor. Se ha detectado la congestión y ningún paquete resultó dañado de ninguna forma. ECN se describe en el RFC 3168. Requiere soporte tanto del host como de los enrutadores, y todavía no se utiliza ampliamente en Internet.

Para obtener más información sobre el conjunto completo de los comportamientos de control de congestión que se implementan en TCP, consulte el RFC 5 681.

6.5.11 El futuro de TCP

Como caballo de batalla de Internet, TCP se ha utilizado para muchas aplicaciones y se ha extendido en el transcurso del tiempo para brindar un buen desempeño a través de un amplio rango de redes. Existen muchas versiones en uso con implementaciones un poco distintas de los algoritmos clásicos que hemos descrito, en especial para el control de la congestión y la robustez ante los ataques. Es probable que TCP continúe evolucionando con Internet. A continuación mencionaremos dos aspectos específicos.

El primero es que TCP no proporciona la semántica de transporte que desean todas las aplicaciones. Por ejemplo, algunas aplicaciones desean enviar mensajes o registros cuyos límites hay que preservar. Otras aplicaciones trabajan con un grupo de conversaciones relacionadas, como un navegador web que transfiere varios objetos del mismo servidor. Existen también otras aplicaciones que desean un mejor control sobre las trayectorias de red que utilizan. TCP con su interfaz de sockets estándar no cumple bien con esas necesidades. En esencia, la aplicación tiene la responsabilidad de lidiar con cualquier problema que TCP no resuelva. Esto ha conducido a propuestas de nuevos protocolos que proporcionen una interfaz ligeramente distinta. Dos ejemplos de esos protocolos son **SCTP** (**Protocolo de Control de Transmisión de Flujo**, del inglés *Stream Control Transmision Protocol*), que se define en el RFC 4 960, y **SST** (**Transporte Estructurado de Flujo**, del inglés *Structured Stream Transport*) (Ford, 2007). Sin embargo, cada vez que alguien propone cambiar algo que ha funcionado tan bien y por tanto tiempo, siempre hay una enorme batalla entre los "usuarios que exigen más características" y los que dicen "si no está roto, no hay que arreglarlo".

El segundo aspecto es el control de la congestión. Tal vez usted piense que éste es un problema resuelto después de nuestras deliberaciones y los mecanismos que se han desarrollado con el tiempo. Pero no es así. La forma de control de congestión en TCP que hemos descrito, y que se utiliza muchos, se basa en las pérdidas de paquetes como señal de congestión. Cuando Padhye y colaboradores (1998) modelaron la tasa de transferencia real de TCP con base en el patrón de diente de sierra, descubrieron que la tasa de pérdida de paquetes debe disminuir con rapidez con base en el aumento de velocidad. Para alcanzar una tasa de transferencia real de 1 Gbps con un tiempo de ida y vuelta de 100 ms y paquetes de 1500 bytes, se puede perder un paquete casi cada 10 minutos. Ésta es una tasa de pérdida de paquetes de 2×10^{-8}, un valor increíblemente pequeño. Es muy poco frecuente como para que sirva como una buena señal de

congestión, y cualquier otra fuente de pérdida (por ejemplo, tasas de errores de transmisión de paquetes de 10^{-7}) puede dominarla con facilidad y, por ende, limitar la tasa de transferencia real.

Esta relación no ha representado un problema en el pasado, pero las redes se están volviendo cada vez más rápidas, lo cual ha orillado a muchas personas a revisar el control de la congestión. Una posibilidad es usar un control de congestión alternativo, en el que la señal no sea ningún tipo de pérdida de paquetes. En la sección 6.2 vimos varios ejemplos. La señal podría ser el tiempo de ida y vuelta, que aumenta cuando la red se congestiona, según se utiliza en FAST TCP (Wei y colaboradores, 2006). Existen también otras metodologías posibles; el tiempo dirá cuál es la mejor.

6.6 ASPECTOS DEL DESEMPEÑO

Los asuntos relacionados con el desempeño son muy importantes en las redes de computadoras. Cuando hay cientos o miles de computadoras conectadas entre sí, son comunes las interacciones complejas, con consecuencias imprevisibles. Con frecuencia, esta complejidad conduce a un desempeño pobre, sin que nadie sepa por qué. En las siguientes secciones examinaremos muchos aspectos relacionados con el desempeño de las redes para ver los tipos de problemas que existen y lo que podemos hacer para resolverlos.

Por desgracia, comprender el desempeño de las redes es más un arte que una ciencia. Existe muy poca teoría al respecto que tenga en realidad alguna utilidad en la práctica. Lo mejor que podemos hacer es dar algunas reglas empíricas derivadas de los tropiezos y ejemplos actuales tomados del mundo real. Hemos postergado de manera intencional este análisis hasta después de estudiar la capa de transporte, debido a que el desempeño que reciben las aplicaciones depende del desempeño combinado de las capas de transporte, de red y de enlace, además de la posibilidad de usar TCP como ejemplo en varios lugares.

En las siguientes secciones analizaremos seis aspectos del desempeño de las redes:

1. Problemas de desempeño.
2. Medición del desempeño de una red.
3. Diseño de hosts para redes rápidas.
4. Procesamiento rápido de los segmentos.
5. Compresión de encabezados.
6. Protocolos para redes de alto desempeño.

Estos aspectos consideran el desempeño de las redes tanto en el host como a través de la misma red, y a medida que las redes aumentan en velocidad y tamaño.

6.6.1 Problemas de desempeño en las redes de computadoras

Algunos problemas de desempeño, como la congestión, se deben a sobrecargas temporales de los recursos. Si repentinamente llega más tráfico a un enrutador del que puede manejar, ocurrirá una congestión y se reducirá el desempeño. Ya estudiamos la congestión con detalle en este capítulo y en el anterior.

El desempeño también se degrada cuando hay un desequilibrio estructural de los recursos. Por ejemplo, si una línea de comunicación de gigabits está conectada a una PC de bajo rendimiento, el pobre host no será capaz de procesar los paquetes de entrada con la suficiente rapidez y se perderán algunos. Tarde o temprano se retransmitirán estos paquetes; lo cual generará un retardo adicional, un desperdicio del ancho de banda y, en general, una reducción en el desempeño.

Las sobrecargas también se pueden desencadenar en forma sincrónica. Por ejemplo, si un segmento contiene un parámetro erróneo (por ejemplo, el puerto al que está destinado), en muchos casos el receptor

devolverá con amabilidad una notificación de error. Ahora considere lo que podría ocurrir si se difundiera un segmento erróneo a 1000 máquinas: cada una podría devolver un mensaje de error. La **tormenta de difusión** resultante podría paralizar la red. UDP sufrió de este problema hasta que se cambió el protocolo ICMP para hacer que los hosts evitaran responder a errores en los segmentos UDP enviados a direcciones de difusión. Las redes inalámbricas deben tener especial cuidado en evitar las respuestas de difusión no verificadas, debido a que la difusión ocurre en forma natural y el ancho de banda inalámbrico es limitado.

Un segundo ejemplo de sobrecarga síncrona es lo que ocurre tras una falla del suministro eléctrico. Al regresar la energía, todas las máquinas arrancan al mismo tiempo. Una secuencia de arranque común podría requerir que la máquina primero acuda a algún servidor (DHCP) para conocer su verdadera identidad, y luego a un servidor de archivos para obtener una copia del sistema operativo. Si cientos de máquinas en un centro de datos hacen todo esto al mismo tiempo, el servidor quizá se colapsaría debido a la carga.

Incluso en ausencia de sobrecargas síncronas y en presencia de suficientes recursos disponibles, puede haber un bajo desempeño debido a la falta de ajuste del sistema. Por ejemplo, si una máquina tiene bastante potencia de CPU y memoria, pero no se ha asignado suficiente memoria como espacio de búfer, el control de flujo reducirá la velocidad de recepción de los segmentos y se limitará el desempeño. Éste fue un problema para muchas conexiones TCP a medida que Internet se hacía más rápida, pero el tamaño predeterminado de la ventana de control de flujo permanecía fija a 64 KB.

Otro asunto relativo a la optimización es el correcto establecimiento de los temporizadores. Cuando se envía un segmento, se establece un temporizador para protegerse contra la pérdida de ese segmento. Si se asigna un valor muy corto al temporizador ocurrirán retransmisiones innecesarias y se obstruirán los cables. Si el valor es demasiado largo, ocurrirán retardos innecesarios después de perder un segmento. Otros parámetros que se pueden optimizar incluyen el tiempo que se deben esperar ciertos datos que se puedan aprovechar para superponer una confirmación de recepción antes de enviarla por separado, y la cantidad de retransmisiones a intentar antes de darse por vencido.

Otro problema de desempeño que ocurre con las aplicaciones de tiempo real, como la transmisión de audio y video, es el jitter. No basta con tener suficiente ancho de banda en promedio para un buen desempeño. También se requieren retardos cortos en la transmisión. Para lograr retardos cortos en forma consistente se requiere un cuidadoso diseño de la carga de la red, un soporte de calidad del servicio en las capas de enlace y de red, o ambos.

6.6.2 Medición del desempeño de las redes

Cuando una red tiene un desempeño pobre, con frecuencia sus usuarios se quejan con los operadores y les exigen mejoras. Para mejorar el desempeño, los operadores deben primero determinar exactamente lo que ocurre. Para averiguar qué está ocurriendo en realidad, deben efectuar mediciones. En esta sección veremos las mediciones de desempeño de las redes. Gran parte del siguiente análisis se basa en el trabajo fundamental de Mogul (1993).

Las mediciones se pueden hacer de muchas maneras y en muchos lugares (tanto físicos como en la pila de protocolos). El tipo de medición más básico es iniciar un temporizador al empezar alguna actividad y medir el tiempo que tarda esa actividad. Por ejemplo, saber cuánto tiempo se requiere para confirmar la recepción de un segmento es una medición clave. Otras mediciones se hacen con contadores que registran la frecuencia con que ocurre un evento (por ejemplo, cantidad de segmentos perdidos). Por último, a menudo nos interesa saber la cantidad de algo, como el número de bytes procesados durante cierto intervalo de tiempo.

La medición del desempeño y los parámetros de una red tiene muchos escollos potenciales. A continuación listaremos aquí algunos de ellos. Cualquier intento sistemático de medir el desempeño de una red debe tener cuidado de evitarlos.

Asegúrese que el tamaño de la muestra sea lo bastante grande

No mida el tiempo para enviar un segmento; mejor repita la medición (por ejemplo, un millón de veces) y obtenga el promedio. Los efectos iniciales, como cuando la NIC 802.16 o el módem de cable obtienen una reservación de ancho de banda después de un periodo de inactividad, pueden reducir la velocidad del primer segmento, además de que el encolamiento introduce la variabilidad. Una muestra grande reducirá la incertidumbre de la media y la desviación estándar medidas. Esta incertidumbre puede calcularse mediante el uso de fórmulas estadísticas estándar.

Asegúrese de que las muestras sean representativas

Lo ideal sería que la secuencia total de un millón de mediciones se repitiera a distintas horas del día y de la semana para ver el efecto de diferentes condiciones de red sobre la cantidad medida. Por ejemplo, las mediciones de congestión son de poco uso si se toman en un momento en el que no hay congestión. A veces los resultados pueden ser contraintuitivos al principio, como la presencia de congestión intensa a las 11:00 a.m. y a la 1:00 p.m., pero sin congestión al mediodía (cuando todos los usuarios están en el almuerzo).

Con las redes inalámbricas, la ubicación es una variable importante debido a la propagación de la señal. Incluso un nodo de medición que se coloque cerca de un cliente inalámbrico tal vez no observe los mismos paquetes que el cliente, debido a diferencias en las antenas. Es mejor tomar medidas desde el cliente inalámbrico que estamos analizando para determinar qué es lo que ve. Si no es posible, tal vez podamos usar técnicas que combinen las mediciones inalámbricas obtenidas en distintos puntos de ventaja para obtener un panorama más completo de lo que está ocurriendo (Mahajan y colaboradores, 2006).

La caché puede arruinar las mediciones

Al repetir una medición varias veces se obtendrá una respuesta inesperadamente rápida si los protocolos usan mecanismos de caché. Por ejemplo, para obtener una página web o buscar un nombre DNS (para encontrar la dirección IP) se puede requerir un intercambio de red la primera vez, y después la respuesta se devuelve desde una caché local sin enviar paquetes a través de la red. Los resultados de dicha medición no tienen en esencia ningún valor (a menos que quiera medir el desempeño de la caché).

Los búferes pueden provocar un efecto similar. Se sabe que algunas pruebas de desempeño de TCP/IP han reportado que UDP puede obtener un desempeño mucho mayor del que la red permite. ¿Cómo ocurre esto? Por lo general, una llamada a UDP devuelve el control tan pronto como el kernel acepta el mensaje y lo agrega a la cola de transmisión. Si hay suficiente espacio en el búfer, sincronizar 1000 llamadas UDP no significa que se hayan enviado todos los datos. La mayoría de ellos pueden estar todavía en el kernel, pero el programa de prueba de desempeño piensa que se han transmitido todos.

Hay que tener cuidado de estar por completo seguros de comprender cómo se pueden colocar los datos en cachés y en búferes como parte de una operación de red.

Asegúrese de que no ocurra nada inesperado durante sus pruebas

Es probable que si realiza mediciones al mismo tiempo en que algún usuario ha decidido llevar a cabo una conferencia de video a través de su red obtenga distintos resultados a los que obtendría si no hubiera una conferencia de video. Es mejor ejecutar las pruebas en una red inactiva y crear la carga completa usted mismo. Aunque esta metodología también presenta algunos obstáculos. Tal vez usted podría pensar que nadie usará la red a las 3:00 A.M., pero esa podría ser precisamente la hora en la que el programa automático de respaldos comienza a copiar todos los discos al sistema de respaldo en cinta

magnética. O tal vez podría haber mucho tráfico para sus maravillosas páginas web desde zonas horarias distantes.

Las redes inalámbricas son retadoras en este aspecto, ya que con frecuencia es imposible separarlas de todas las fuentes de interferencia. Incluso aunque no haya otras redes inalámbricas cercanas que envíen tráfico, alguien podría usar el microondas para cocinar unas palomitas y provocar, sin querer, una advertencia que degrade el desempeño de la red 802.11. Por estas razones, es conveniente supervisar la actividad de la red en general para al menos saber cuando ocurre algo inesperado.

Tenga cuidado al usar un reloj de grandes intervalos

La función de los relojes de computadora es sumar uno a un contador, a intervalos regulares. Por ejemplo, un temporizador de milisegundos suma 1 al contador cada 1 mseg. Es posible usar dicho temporizador para medir un evento que tarda menos de 1 mseg, pero requiere cierto cuidado. Claro que algunas computadoras tienen relojes más precisos, pero también es cierto que siempre hay eventos más cortos qué medir. Observe que los relojes no son siempre tan precisos como la precisión con la que se devuelve el tiempo al momento de leerlos.

Por ejemplo, para medir el tiempo de creación de una conexión TCP, hay que leer el reloj del sistema (digamos, en milisegundos) al entrar en el código de capa de transporte y una vez más al salir. Si el tiempo real de establecimiento de la conexión es de 300 μseg, la diferencia entre las dos lecturas será 0 o 1, ambas cifras equivocadas. Pero si la medición se repite un millón de veces, se suman todas las mediciones y se dividen entre un millón, el tiempo medio tendrá una precisión del orden de menos de 1 μseg.

Tenga cuidado con la extrapolación de los resultados

Suponga que hace mediciones con cargas de red simuladas que van desde 0 (inactividad) a 0.4 (40% de la capacidad), Por ejemplo, el tiempo de respuesta para enviar un paquete de voz sobre IP a través de una red 802.11 podría ser como se indica mediante los puntos de datos y la línea continua que pasa a través de ellos en la figura 6-49. Puede ser tentador extrapolar linealmente, como lo indica la línea punteada. Sin embargo, muchos resultados de encolamiento comprenden un factor de $1/(1 - \rho)$, donde ρ es la carga, por lo que los valores verdaderos pueden parecerse más a la línea punteada, que se eleva más rápido que en forma lineal cuando hay mucha carga. En resumen, hay que cuidarse de los efectos de contención que se vuelven mucho más pronunciados cuando hay mucha carga.

6.6.3 Diseño de hosts para redes rápidas

Las mediciones y los ajustes pueden mejorar con frecuencia el desempeño de una manera considerable, pero no pueden sustituir un buen diseño en primer lugar. Una red mal diseñada se puede mejorar sólo hasta cierto punto. Más allá de eso, hay que rediseñarla desde cero.

En esta sección presentaremos algunas reglas empíricas para la implementación del software de los protocolos de red en los hosts. Para nuestra sorpresa, la experiencia demuestra que éste es un cuello de botella común para el desempeño en redes que de otra manera serían rápidas, por dos razones. Primera, las tarjetas NIC (Tarjetas de Interfaz de Red) y los enrutadores ya se han diseñado (con soporte de hardware) para operar a la "velocidad del alambre". Esto significa que pueden procesar paquetes con tanta rapidez como éstos pueden llegar al enlace. Segunda, el desempeño relevante es lo que obtienen las aplicaciones. No es la capacidad del enlace, sino la tasa de transferencia real y el retardo después del procesamiento de las capas de red y de transporte.

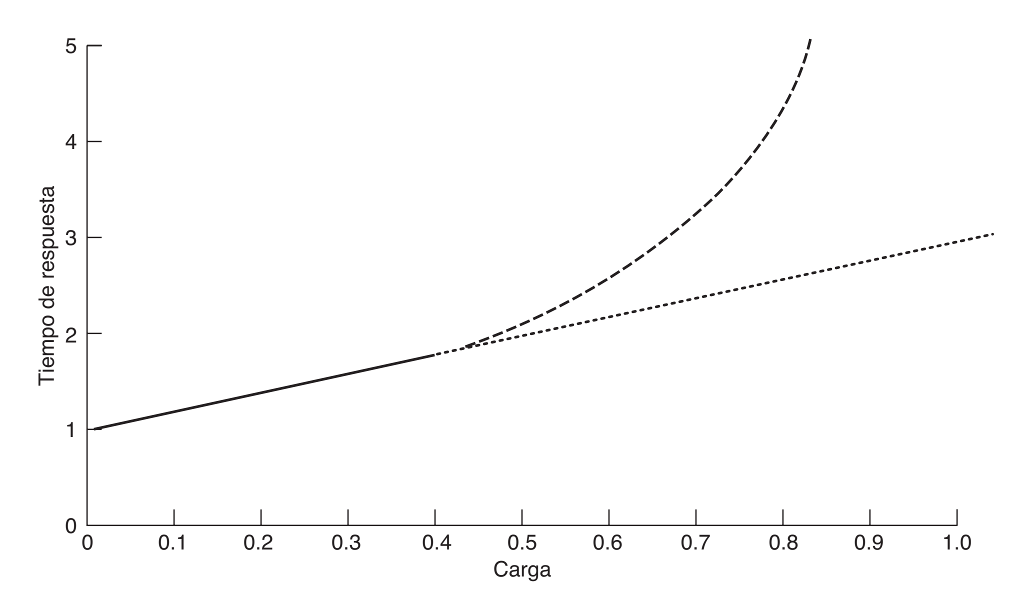


Figura 6-49. La respuesta como una función de la carga.

Al reducir las sobrecargas de software aumenta el desempeño, puesto que se incrementa la tasa de transferencia real y se reduce el retardo. También es posible reducir la energía que se invierte en el trabajo en red, lo cual es una consideración importante para las computadoras móviles. La mayor parte de estas ideas han sido del conocimiento común de los diseñadores de redes durante años. Mogul fue el primero en postularlas en forma explícita (1993); nuestro estudio sigue en gran parte la secuencia del suyo. Otra fuente relevante es Metcalfe (1993).

La velocidad del host es más importante que la velocidad de la red

La amplia experiencia ha demostrado que en casi todas las redes rápidas, la sobrecarga de los sistemas operativos y protocolos domina el tiempo real en el alambre. Por ejemplo, en teoría, el tiempo mínimo de una RPC en una red Ethernet de 1 Gbps es de 1 μseg, lo cual corresponde a una solicitud mínima (512 bytes) seguida de una respuesta mínima (512 bytes). En la práctica, reducir la sobrecarga de software y hacer que el tiempo de la RPC esté lo más cerca posible de 1 μseg es un logro considerable. Esto ocurre raras veces en la práctica.

Asimismo, el mayor problema al operar a 1 Gbps es llevar comúnmente los bits desde el búfer del usuario hasta la red con una velocidad suficiente y lograr que el host receptor los procese tan rápido como lleguen. Si se duplica la velocidad del host (CPU y memoria), con frecuencia casi se puede duplicar la tasa de transferencia real. En muchos casos no tiene efecto duplicar la capacidad de la red si el cuello de botella está en los hosts.

Reducir el conteo de paquetes para disminuir la sobrecarga

Cada segmento tiene cierta cantidad de sobrecarga (por ejemplo, el encabezado) al igual que datos (la carga útil). Se requiere ancho de banda para ambos componentes. También se requiere el procesamiento para ambos componentes (por ejemplo, procesar el encabezado y realizar la suma de verificación). Al enviar 1 millón de bytes, el costo de los datos es el mismo, sin importar cuál sea el tamaño del segmento. Sin embargo, utilizar segmentos de 128 bytes implica 32 veces más sobrecarga por segmento que usar

segmentos de 4 KB. Las sobrecargas de ancho de banda y de procesamiento se acumulan con rapidez para reducir la tasa de transferencia real.

La sobrecarga por paquete en las capas inferiores amplifica este efecto. Cada paquete que llega provoca una nueva interrupción si el host está manteniendo el ritmo. En un procesador moderno con canalización, cada interrupción rompe la canalización de la CPU, interfiere con la caché, requiere un cambio en el contexto de administración de la memoria, invalida la tabla de predicción de bifurcación y obliga a almacenar una cantidad considerable de registros de CPU. Así, una reducción de *n* veces en los segmentos enviados reduce la sobrecarga de la interrupción y de los paquetes en un factor de *n*.

Podríamos decir que tanto las personas como las computadoras son malas para realizar multitarea. Esta observación se relaciona con el deseo de enviar paquetes de MTU cuyo tamaño sea el mayor posible como para pasar por la trayectoria de red sin necesidad de fragmentación. Los mecanismos tales como el algoritmo de Nagle y la solución de Clark son también intentos por evitar enviar paquetes pequeños.

Minimizar las copias de datos

La forma más simple y directa de implementar una pila de protocolos en capas es usar un módulo para cada capa. Por desgracia, esto conduce a la copia de datos (o por lo menos, acceder a los datos en varias pasadas) mientras cada capa realiza su propio trabajo. Por ejemplo, después de que la NIC recibe un paquete, por lo general se copia a un búfer del kernel. De ahí se copia a un búfer en la capa de red para que ésta lo procese, después a un búfer de la capa de transporte para que ésta lo procese y, por último, al proceso de aplicación receptor. Es común que un paquete entrante se copie tres o cuatro veces antes de entregar el segmento incluido en él.

Toda esta copia puede degradar el desempeño en forma considerable, ya que las operaciones en memoria son hasta 10 veces más lentas que las instrucciones de registro a registro. Por ejemplo, si sólo el 20% de las instrucciones van a memoria (por ejemplo, fallas en la caché), lo cual es probable cuando se copian los paquetes entrantes, el tiempo promedio de ejecución de instrucciones se alarga por un factor de 2.8 ($0.8 \times 1 + 0.2 \times 10$). La asistencia del hardware no será de utilidad aquí. El problema es que el sistema operativo realiza muchas copias.

Un sistema operativo ingenioso minimizará el copiado mediante la combinación del procesamiento de varias capas. Por ejemplo, es común que TCP e IP se implementen juntos (como "TCP/IP"), de modo que no es necesario copiar la carga útil del paquete a medida que el procesamiento cambia de la capa de red a la de transporte. Otro truco común es realizar varias operaciones dentro de una capa en una sola pasada sobre los datos. Por ejemplo, a menudo las sumas de verificación se calculan mientras se copian los datos (cuando hay que copiarlos) y la suma de verificación recién calculada se adjunta al final.

Minimizar las conmutaciones de contexto

Una regla relacionada es que las conmutaciones de contexto (por ejemplo, del modo de kernel al modo de usuario) son mortales. Tienen los mismos inconvenientes que las interrupciones y las copias combinadas. Este costo es la razón por la que los protocolos de transporte se implementan casi siempre en el kernel. Así como se reduce el conteo de paquetes, también se pueden reducir las conmutaciones de contexto al hacer que el procedimiento de biblioteca que envía datos use un búfer interno hasta que tenga una cantidad considerable de éstos. Asimismo, en el lado receptor los pequeños segmentos entrantes se deben agrupar y pasar al usuario de un solo golpe, en vez de pasarlos por separado para minimizar las conmutaciones de contexto.

En el mejor de los casos, un paquete entrante provoca una conmutación de contexto del usuario al kernel, y después una conmutación al proceso receptor para proporcionarle los datos que acaban de

llegar. Por desgracia, en algunos sistemas operativos ocurren conmutaciones de contexto adicionales. Por ejemplo, si el administrador de la red se ejecuta como un proceso especial en espacio de usuario, al llegar un paquete es probable que provoque una conmutación de contexto del usuario actual al kernel, después otra del kernel al administrador de red, seguida de otra de regreso al kernel y, por último, una del kernel al proceso receptor. Esta secuencia se muestra en la figura 6-50. Todas estas conmutaciones de contexto en cada paquete desperdician mucho tiempo de CPU y pueden tener un efecto devastador sobre el desempeño de la red.

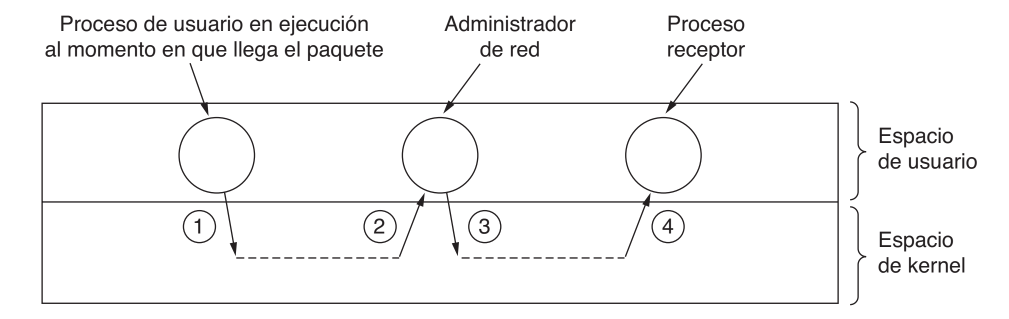


Figura 6-50. Cuatro conmutaciones de contexto para manejar un paquete con un administrador de red en el espacio de usuario.

Evitar la congestión es mejor que recuperarse de ella

La vieja máxima de que más vale prevenir que lamentar sí aplica a la congestión en las redes. Al congestionarse una red se pierden paquetes, se desperdicia ancho de banda, se introducen retardos inútiles y otras cosas. Todos estos costos son innecesarios; la recuperación requiere tiempo y paciencia. Es mejor que no ocurra en primer lugar. Evitar la congestión es como recibir una vacuna: duele un poco en el momento, pero evita algo que sería mucho más doloroso en el futuro.

Evitar expiraciones del temporizador

Los temporizadores son necesarios en las redes, pero hay que usarlos con cuidado y se deben reducir al mínimo los tiempos de expiración. Al expirar un temporizador, lo común es que se repita una acción. Si en realidad es necesario repetir la acción, que así sea, pero su repetición innecesaria es un desperdicio.

La manera de evitar el trabajo extra es tener cuidado de que los intervalos del temporizador sean más bien conservadores. Un temporizador que tarda demasiado en expirar agrega una pequeña cantidad de retardo extra a una conexión en el caso (improbable) de que se pierda un segmento. Un temporizador que expira cuando no debería consume recursos del host, desperdicia ancho de banda e impone una carga adicional tal vez a docenas de enrutadores sin una buena razón.

6.6.4 Procesamiento rápido de segmentos

Ahora que hemos cubierto las reglas generales, vamos a analizar algunos métodos específicos para agilizar el procesamiento de segmentos. Para obtener más información, consulte a Clark y colaboradores (1989), y también a Chase y colaboradores (2001).

La sobrecarga de procesamiento de los segmentos tiene dos componentes: la sobrecarga por segmento y la sobrecarga por byte. Ambas deben combatirse. La clave para el procesamiento rápido de los

segmentos es separar el caso normal de éxito (transferencia de datos en un solo sentido) y manejarlo como caso especial. Muchos protocolos tienden a enfatizar lo que deben hacer cuando algo sale mal (por ejemplo, cuando se pierde un paquete), pero para que los protocolos operen con rapidez, el diseñador debería concentrarse en minimizar el tiempo de procesamiento cuando todo sale bien. Minimizar el tiempo de procesamiento cuando ocurre un error es algo secundario.

Aunque se necesita una secuencia de segmentos especiales para entrar en el estado ESTABLISHED (establecido), una vez ahí el procesamiento de los segmentos es directo hasta que un lado empieza a cerrar la conexión. Comencemos por examinar el lado emisor en el estado ESTABLISHED cuando hay datos por transmitir. Por cuestión de claridad, vamos a suponer aquí que la entidad de transporte está en el kernel, aunque se aplican los mismos conceptos si es un proceso de espacio de usuario o una biblioteca dentro del proceso emisor. En la figura 6-51, el proceso emisor causa una interrupción en el kernel para ejecutar SEND. Lo primero que hace la entidad de transporte es probar si éste es el caso normal: el estado es ESTABLISHED, ningún lado está tratando de cerrar la conexión, se está enviando un segmento normal (es decir, no fuera de banda) completo, y hay suficiente espacio disponible de ventana en el receptor. Si se cumplen todas las condiciones, no se requieren pruebas adicionales y se puede tomar la trayectoria rápida a través de la entidad de transporte emisora. Por lo general, esta ruta se toma la mayoría de las veces.

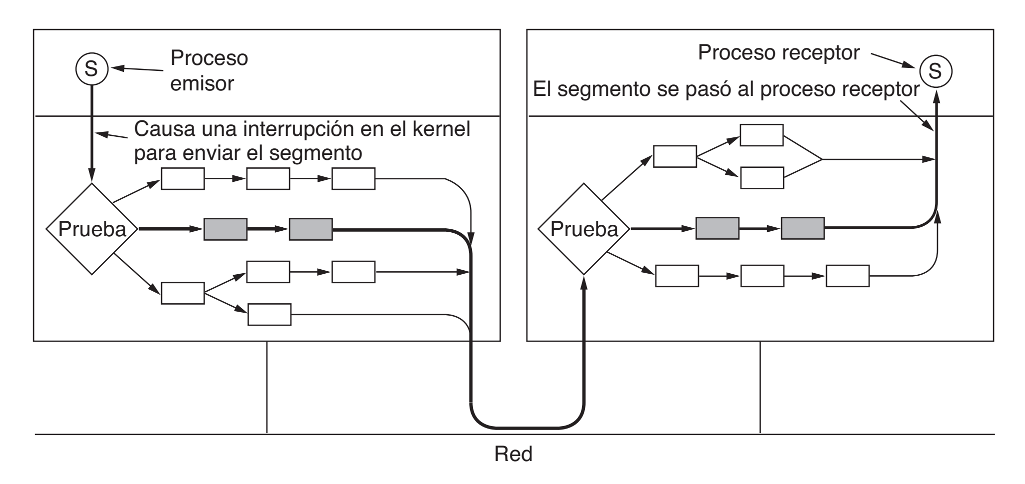


Figura 6-51. La trayectoria rápida del emisor al receptor se muestra con una línea gruesa. Los pasos de procesamiento en esta trayectoria están sombreados.

En el caso normal, los encabezados de los segmentos de datos consecutivos son casi los mismos. Para aprovechar este hecho, se almacena un encabezado prototipo en la entidad de transporte. Al principio de la trayectoria rápida, el encabezado se copia lo más rápido posible en un búfer de trabajo, palabra por palabra. Los campos que cambian de un segmento a otro se sobrescriben en el búfer. Con frecuencia, estos campos se deducen fácilmente de las variables de estado, como el siguiente número de secuencia. A continuación, se pasan a la capa de red un apuntador al encabezado completo del segmento más un apuntador a los datos de usuario. Aquí se puede seguir la misma estrategia (no se muestra en la figura 6-51). Por último, la capa de red entrega el paquete resultante a la capa de enlace de datos para su transmisión.

Como ejemplo del funcionamiento de este principio en la práctica, consideremos TCP/IP. En la figura 6-52(a) se muestra el encabezado TCP. Los campos que son iguales entre segmentos consecutivos de un flujo en un solo sentido aparecen sombreados. Todo lo que tiene que hacer la entidad de transporte emisora es copiar las cinco palabras del encabezado prototipo en el búfer de salida, llenar el siguiente número de secuencia (que copia de una palabra en la memoria), calcular la suma de verificación e incrementar el

número de secuencia en la memoria. Entonces puede entregar el encabezado y los datos a un procedimiento IP especial para enviar un segmento máximo normal. El IP entonces copia su encabezado prototipo de cinco palabras [vea la figura 6-52(b)] en el búfer, llena el campo de *Identificación* y calcula su suma de verificación. Ahora el paquete ya está listo para transmitirse.

Veamos ahora el rápido procesamiento de la trayectoria del lado receptor de la figura 6-51. El paso 1 es localizar el registro de conexión del segmento entrante. En TCP, el registro de conexión se puede almacenar en una tabla de hash en la que alguna función simple de las dos direcciones IP y los dos puertos es la clave. Una vez localizado el registro de conexión, hay que comparar ambas direcciones y ambos puertos para verificar que se haya encontrado el registro correcto.

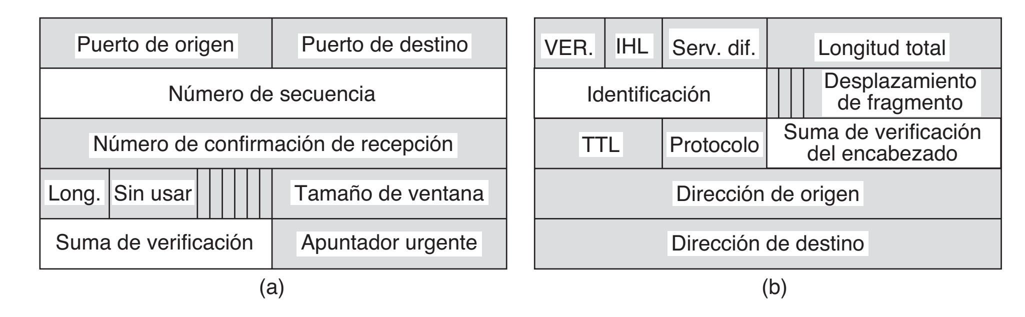


Figura 6-52. (a) Encabezado TCP. (b) Encabezado IP. En ambos casos se toman del prototipo sin cambios.

Una optimización que con frecuencia acelera aún más la búsqueda del registro de conexión es la de mantener un apuntador al último registro usado y probar ese primero. Clark y colaboradores (1989) probaron esto y observaron una tasa de éxito mayor al 90%.

A continuación se verifica el segmento para ver si es normal: el estado es ESTABLISHED, ninguno de los dos lados está tratando de cerrar la conexión, el segmento está completo, no hay banderas especiales activadas y el número de secuencia es el esperado. Estas pruebas se llevan apenas unas cuantas instrucciones. Si se cumplen todas las condiciones, se invoca un procedimiento TCP especial de trayectoria rápida.

La trayectoria rápida actualiza el registro de la conexión y copia los datos al espacio de usuario. Mientras copia, el procedimiento también calcula la suma de verificación, con lo cual elimina un paso extra sobre los datos. Si la suma de verificación es correcta, se actualiza el registro de conexión y se devuelve una confirmación de recepción. El esquema general de hacer primero una comprobación rápida para ver si el encabezado es el esperado y tener después un procedimiento especial para manejar ese caso se llama **predicción de encabezado**. Muchas implementaciones de TCP lo usan. Cuando esta optimización y todas las demás estudiadas en este capítulo se usan juntas, es posible conseguir que el TCP opere 90% de la velocidad de una copia local de memoria a memoria, suponiendo que la red misma sea lo bastante rápida.

Otras dos áreas en las que es posible obtener mejoras sustanciales del desempeño son el manejo de búferes y la administración de los temporizadores. El aspecto importante del manejo de búferes es evitar el copiado innecesario, como se explicó antes. La administración de los temporizadores es importante porque casi ninguno de los temporizadores expira. Se ajustan para protegerse contra pérdidas de segmentos, pero la mayoría de los segmentos llegan correctamente, al igual que sus confirmaciones de recepción. Por tanto, es importante optimizar el manejo de los temporizadores para el caso en que casi nunca expiren.

Un esquema común consiste en usar una lista enlazada de eventos de temporizador, ordenada por hora de expiración. La entrada inicial contiene un contador que indica la cantidad de pulsos de reloj que faltan para la expiración. Cada entrada subsecuente contiene un contador que indica a cuántos pulsos se

encuentra después de la entrada anterior. Por lo tanto, si los temporizadores expiran en 3, 10 y 12 pulsos, respectivamente, los tres contadores son de 3, 7 y 2.

Después de cada pulso de reloj, se decrementa el contador del encabezado inicial. Cuando llega a cero, su evento se procesa y el siguiente elemento de la lista es ahora el inicial. No hay que cambiar su contador. De esta manera, insertar y eliminar temporizadores son operaciones costosas, con tiempos de ejecución proporcionales a la longitud de la lista.

Podemos usar un método mucho más eficiente si el intervalo máximo del temporizador está limitado y se conoce por adelantado. Aquí se puede utilizar un arreglo llamado **rueda de temporización**, como se muestra en la figura 6-53. Cada ranura corresponde a un pulso de reloj. El tiempo actual mostrado en la figura es $T = 4$. Los temporizadores se programan para expirar a 3, 10 y 12 pulsos más adelante. Si se establece un temporizador nuevo para expirar en siete pulsos, simplemente se crea una entrada en la ranura 11. Del mismo modo, si hay que cancelar el temporizador establecido para $T + 10$, es necesario examinar la lista que comienza en la ranura 14 para eliminar la entrada pertinente. Observe que el arreglo de la figura 6-53 no puede manejar temporizadores más allá de $T + 15$.

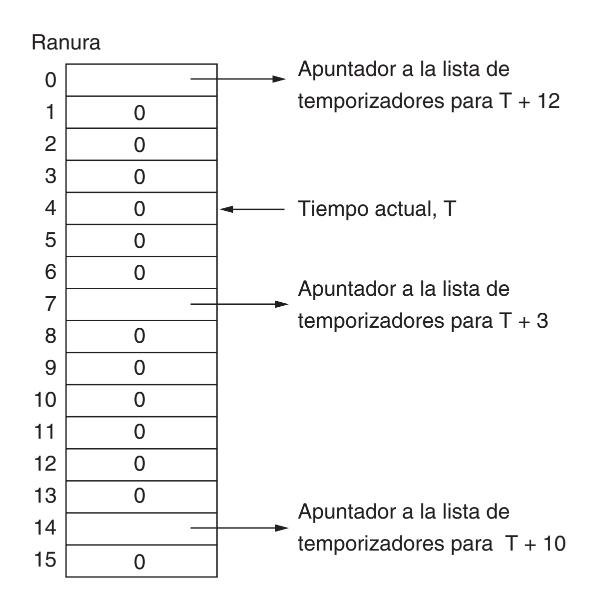


Figura 6-53. Una rueda de temporización.

Cuando el reloj pulsa, el apuntador de tiempo actual avanza una ranura (en forma circular). Si la entrada a la que ahora se apunta es diferente a cero, todos sus temporizadores se procesan. Encontrará muchas variaciones de la idea básica en Varghese y Lauck (1987).

6.6.5 Compresión de encabezado

Hemos estado analizando las redes rápidas por mucho tiempo. Es tiempo de ver otras redes. Vamos a considerar ahora el desempeño en las redes inalámbricas y otras redes en donde el ancho de banda es limitado. Reducir la sobrecarga de software puede ayudar a las computadoras móviles a operar con más eficiencia, pero no hace nada por mejorar el desempeño cuando los enlaces de la red son el “cuello de botella”.

Para usar bien el ancho de banda, hay que transportar los encabezados de protocolo y las cargas útiles con el mínimo de bits. Para las cargas útiles esto significa usar codificaciones compactas de información, como las imágenes en formato JPEG en vez de mapas de bits, o los formatos de documentos tales como PDF, que incluyen compresión. También significa el uso de mecanismos de caché a nivel de aplicación, como las cachés web que reducen las transferencias en primer lugar.

¿Qué hay para los encabezados de protocolo? En la capa de enlace, por lo general los encabezados para las redes inalámbricas son compactos, debido a que se diseñaron teniendo en cuenta el escaso ancho de banda. Por ejemplo, los encabezados 802.16 tienen identificadores cortos de conversión en vez de direcciones más largas. Pero los protocolos de capa superior tales como IP, TCP y UDP vienen en una sola versión para todas las capas de enlace; además no están diseñados con encabezados compactos. De hecho, el procesamiento modernizado para reducir la sobrecarga de software a menudo conduce a encabezados que no son tan compactos como podrían serlo de otro modo (por ejemplo, IPv6 tiene encabezados empaquetados de una manera más flexible que IPv4).

Los encabezados de las capas superiores pueden tener un impacto considerable en el desempeño. Por ejemplo, considere los datos de voz sobre IP que se transportan con la combinación de IP, UDP y RTP. Estos protocolos requieren 40 bytes de encabezado (20 para IPv4, 8 para UDP y 12 para RTP). Con IPv6, la situación es aún peor: 60 bytes, incluyendo el encabezado IPv6 de 40 bytes. Los encabezados pueden llegar a representar la mayoría de los datos transmitidos y consumir más de la mitad del ancho de banda.

La **compresión de encabezados** se utiliza para reducir el ancho de banda que ocupan los encabezados de los protocolos de capas superiores en los enlaces. Se utilizan esquemas diseñados de manera especial, en vez de métodos de propósito general. Esto se debe a que los encabezados son cortos, por lo que no se comprimen bien por separado; además, para descomprimirlos es necesario recibir todos los datos anteriores. Si se pierde un paquete, esto no será posible.

La compresión de encabezados obtiene ventajas considerables gracias a que se conoce el formato del protocolo. Van Jacobson (1990) diseñó uno de los primeros esquemas para comprimir encabezados TCP/IP a través de enlaces seriales lentos. Este esquema puede comprimir un encabezado TCP/IP común de 40 bytes hasta un promedio de 3 bytes. En la figura 6.52 se da una pista en relación con el truco de este método. Muchos de los campos de encabezado no cambian de un paquete a otro. Por ejemplo, no hay necesidad de enviar el mismo TTL de IP, o los mismos números de puerto TCP en todos y cada uno de los paquetes. Se pueden emitir en el lado emisor del enlace y llenarlos después en el lado receptor.

Asimismo, otros campos cambian de una manera predecible. Por ejemplo, a menos que haya pérdidas, el número de secuencia TCP avanza con los datos. En estos casos, el receptor puede predecir el valor probable. El número real sólo se necesita transportar cuando difiere de lo esperado. Aún así, se puede transportar como un pequeño cambio a partir del valor anterior, como cuando el número de confirmación de recepción aumenta cuando se reciben nuevos datos en el sentido inverso.

Con la compresión de encabezados, es posible tener encabezados simples en protocolos de capas superiores y codificaciones compactas a través de enlaces con poco ancho de banda. **ROHC** (**Compresión Robusta de Encabezados**, del inglés *RObust Header Compression*) es una versión moderna de compresión de encabezados que se define como un marco de trabajo en el RFC 5795. Está diseñada para tolerar la pérdida que puede ocurrir en los enlaces inalámbricos. Hay un perfil para cada conjunto de protocolos que se van a comprimir, como IP/UDP/RTP. Los encabezados comprimidos se transportan mediante la referencia a un contexto, que en esencia es una conexión; los campos de los encabezados se pueden predecir con facilidad para los paquetes de la misma conexión, pero no para los paquetes de distintas conexiones. En una operación típica, ROHC reduce los encabezados IP/UDP/RTP de 40 bytes hasta un valor entre 1 y 3 bytes.

Aunque la compresión de encabezados se enfoca principalmente en reducir las necesidades de ancho de banda, también puede ser útil para reducir el retardo. Éste se compone del retardo de propagación, que es fijo para una trayectoria de red dada, y del retardo de transmisión que depende del ancho de banda y

la cantidad de datos a enviar. Por ejemplo, un enlace de 1 Mbps envía 1 bit en 1 μseg. En el caso de los medios a través de redes inalámbricas, la red es relativamente lenta por lo que el retardo de transmisión puede ser un factor importante en el retardo en general; además es importante tener un retardo bajo de manera consistente para la calidad del servicio.

La compresión de encabezados puede ayudar a reducir la cantidad de datos que se envían y, por ende, reduce el retardo de transmisión. Es posible obtener el mismo efecto al enviar paquetes más pequeños. En este caso se hace un canje entre un aumento en la sobrecarga del software para obtener un menor retardo de transmisión. Observe que otra fuente potencial de retardo es el retardo de encolamiento para acceder al enlace inalámbrico. Este tipo de retardo también puede ser considerable, ya que con frecuencia los enlaces inalámbricos se utilizan mucho como el recurso limitado en una red. En este caso, el enlace inalámbrico debe tener mecanismos de calidad del servicio que proporcionen un retardo bajo a la hora de enviar paquetes en tiempo real. No basta sólo con la compresión de encabezados.

6.6.6 Protocolos para redes de alto desempeño

Desde la década de 1990 han existido las redes de gigabits que transmiten datos a través de largas distancias. Debido a la combinación de una red rápida, o “canal grueso”, y un retardo grande, a estas redes se les conoce como **redes long fat**. Cuando surgieron estas redes, la primera reacción de la gente fue usar en ellas los protocolos existentes, pero pronto surgieron varios problemas. En esta sección estudiaremos algunos problemas relacionados con el aumento en la velocidad y el retardo de los protocolos de red.

El primer problema es que muchos protocolos usan números de secuencia de 32 bits. Cuando empezó Internet, las líneas entre enrutadores eran en su mayoría líneas rentadas de 56 kbps, por lo que un host que transmitiera a toda velocidad tomaba alrededor de una semana para dar vuelta a los números de secuencia. Para los diseñadores de TCP, 2^{32} era una aproximación muy buena al infinito porque había poco riesgo de que los paquetes viejos deambularan por la red una semana después de su transmisión. Con la Ethernet de 10 Mpbs, el tiempo para dar vuelta a los números de secuencia se redujo a 57 minutos; mucho más corto, pero aún manejable. Con una Ethernet de 1 Gbps que descarga sus datos hacia Internet, el tiempo aproximado para dar vuelta a los números de secuencia es de 34 segundos, un valor muy por debajo de los 120 segundos de tiempo de vida máximo de un paquete en Internet. De pronto, 2^{32} ya no se considera una buena aproximación al infinito, puesto que un emisor veloz puede recorrer el espacio de secuencia sin importar que aún existan paquetes viejos en la red.

El problema es que muchos diseñadores de protocolos sólo supusieron tácitamente que el tiempo requerido para consumir el espacio de secuencias completo excedería por mucho el tiempo de vida máximo de los paquetes. En consecuencia, no había necesidad de preocuparse siquiera por el problema de que existieran todavía duplicados viejos al momento en que se diera vuelta a los números de secuencia. A velocidades de gigabits, ese supuesto implícito fracasa. Por fortuna, se descubrió que era posible extender el número de secuencia efectivo si se consideraba la estampa de tiempo, que se puede transportar como una opción en el encabezado TCP de cada paquete como los bits de mayor orden. Este mecanismo se conoce como PAWS (Protección contra el Reinicio de Números de Secuencia) y se describe en el RFC 1323.

Un segundo problema es el de tener que incrementar de manera considerable el tamaño de la ventana de control de flujo. Por ejemplo, considere el ejemplo en el que se envía una ráfaga de 64 KB de datos de San Diego a Boston para llenar el búfer de 64 KB del receptor. Suponga que el enlace es de 1 Gbps y que el retardo en un solo sentido de la fibra óptica, que puede operar a la velocidad de la luz, es de 20 mseg. En un principio, en $t = 0$, el canal está vacío como se muestra en la figura 6-54(a). Tan sólo 500 μseg más tarde, en la figura 6-54(b), todos los segmentos se encuentran ya viajando por la fibra. El segmento principal estará ahora en alguna parte cerca de Brawley, todavía muy dentro del sur de California. Sin embargo, el transmisor debe detenerse hasta obtener una actualización de ventana.

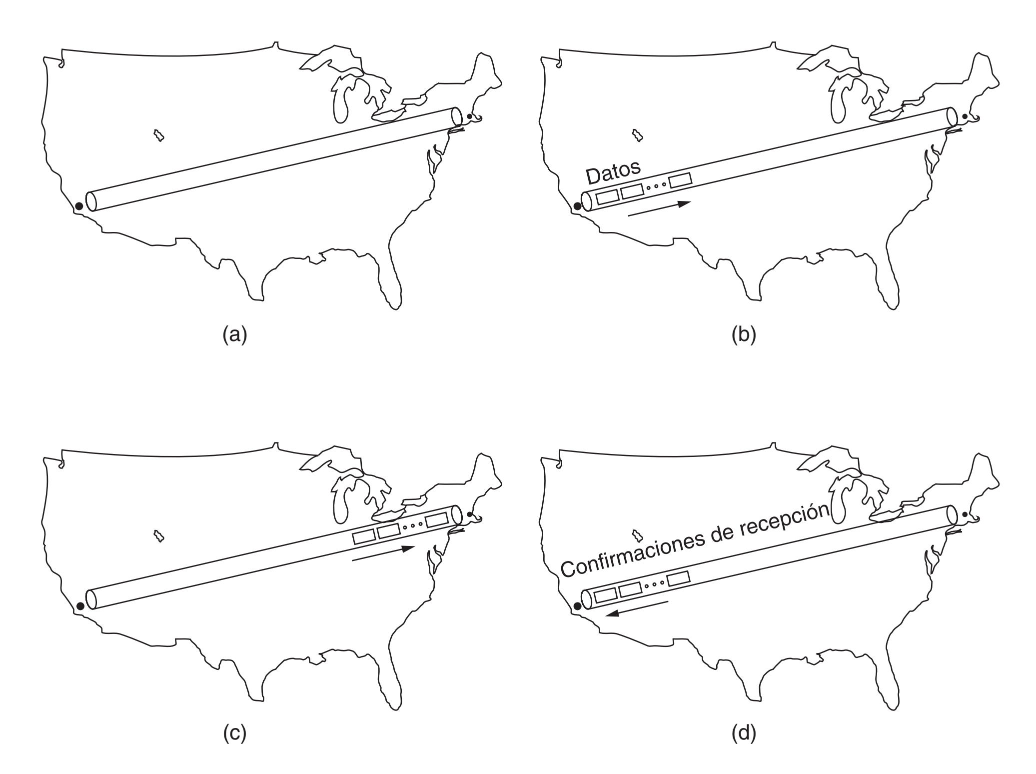


Figura 6-54. El estado de transmitir 1 Mbit de San Diego a Boston. (a) Cuando $t = 0$. (b) Después de 500 µseg. (c) Después de 20 mseg. (d) Después de 40 mseg.

Después de 20 mseg, el segmento principal llega a Boston, como se muestra en la figura 6.54(c), y se confirma su recepción. Finalmente, 40 mseg después de iniciar, la primera confirmación de recepción regresa al emisor y se puede transmitir la segunda ráfaga. Como la línea de transmisión se usó durante 1.25 mseg de los 100 disponibles, la eficiencia aproximada es de 1.25%. Esta situación es común de los protocolos antiguos que operan en líneas de gigabits.

Una cantidad útil a tener en cuenta al momento de analizar el desempeño de una red es el **producto ancho de banda-retardo**, que se obtiene al multiplicar el ancho de banda (en bits/seg) por el tiempo de retardo de ida y vuelta (en segundos). El producto es la capacidad del canal, desde el emisor hasta el receptor y en sentido inverso (en bits).

Para el ejemplo de la figura 6-54, el producto ancho de banda-retardo es de 40 millones de bits. En otras palabras, el emisor tendría que transmitir una ráfaga de 40 millones de bits para seguir operando a toda velocidad hasta que regresara la primera confirmación de recepción. Se requieren todos estos bits para llenar el canal (en ambos sentidos). Esto explica por qué una ráfaga de medio millón de bits sólo obtiene una eficiencia de 1.25%; constituye tan sólo el 1.25% de la capacidad del canal.

La conclusión que podemos deducir de todo esto es que para un buen desempeño, la ventana del receptor debe ser por lo menos tan grande como el producto ancho de banda-retardo, y de preferencia algo más grande ya que el receptor tal vez no responda al instante. Para una línea de gigabits transcontinental, se requieren por lo menos 5 MB.

Un tercer problema relacionado es que los esquemas simples de retransmisión, como el protocolo de retroceso *n*, tienen un desempeño pobre en las líneas con un producto ancho de banda-retardo grande. Por ejemplo, considere un enlace transcontinental de 1 Gbps con un tiempo de transmisión de ida y vuelta de 40 mseg. Un emisor puede transmitir 5 MB en un viaje de ida y vuelta. Si se detecta un error, pasarán 40

mseg antes de que el emisor se entere. Si se usa el retroceso *n*, el emisor tendrá que retransmitir no sólo el paquete erróneo, sino también los 5 MB de paquetes que llegaron después. En definitiva, éste es un desperdicio masivo de recursos. Se requieren protocolos más complejos como repetición selectiva.

Un cuarto problema es que las líneas de gigabits son fundamentalmente diferentes de las líneas de megabits en cuanto a que las líneas largas de gigabits están limitadas por el retardo en vez del ancho de banda. En la figura 6-55 mostramos el tiempo requerido para transferir un archivo de 1 Mbit por 4 000 km con varias velocidades de transmisión. A velocidades de hasta 1 Mbps, el tiempo de transmisión está dominado por la tasa a la que se pueden enviar los bits. A 1 Gbps, el retardo de viaje de ida y vuelta de 40 mseg domina el 1 mseg que se requiere para colocar los bits en la fibra. Si se aumenta más el ancho de banda, el efecto es muy insignificante.

La figura 6-55 tiene implicaciones desafortunadas para los protocolos de red. Indica que los protocolos de parada y espera, como el RPC, tienen un límite superior inherente en su desempeño. Este límite lo establece la velocidad de la luz. Ningún progreso tecnológico en la óptica mejorará la situación (aunque si se descubrieran nuevas leyes de la física, ayudaría mucho). A menos que se pueda encontrar algún otro uso para una línea de un gigabit mientras un host espera una respuesta, esta línea no será mejor que una línea de un megabit; simplemente será más costosa.

Un quinto problema es que las velocidades de comunicación han mejorado con mucha mayor rapidez que las velocidades de las computadoras. (Nota a los ingenieros en computación: ¡salgan a darles una paliza a esos ingenieros en comunicaciones! Contamos con ustedes). En la década de 1970, ARPANET operaba a 56 kbps y tenía computadoras que operaban casi a 1 MIPS. Compare estas cifras con las computadoras de 1000 MIPS que intercambian paquetes a través de una línea de 1 Gbps. El número de instrucciones por byte se ha reducido por un factor mayor de 10. Los números exactos son debatibles dependiendo de las fechas y los escenarios, pero la conclusión es ésta: hay menos tiempo disponible para el procesamiento de los protocolos del que solía haber, por lo que debemos simplificarlos más.

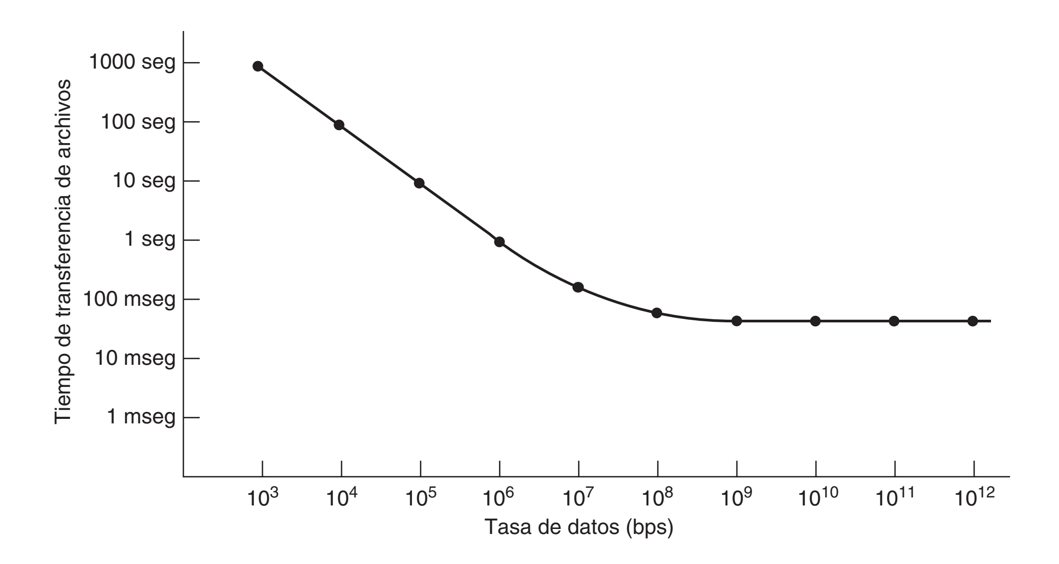


Figura 6-55. Tiempo para transferir y confirmar la recepción de un archivo de 1 Mbit a través de una línea de 4 000 km.

Ahora dejemos de hablar sobre los problemas y veamos cómo resolverlos. El principio básico que deben aprender de memoria todos los diseñadores de redes de alta velocidad es:

Diseñar con miras en la velocidad, no en la optimización del ancho de banda.

Los protocolos viejos con frecuencia se diseñaban para tratar de reducir al mínimo la cantidad de bits en el alambre; por lo general se usaban campos pequeños y los bits se empaquetaban en bytes y palabras. Esta cuestión aún es válida para las redes inalámbricas, pero no para las redes de gigabit. El procesamiento del protocolo es el problema, por lo que los protocolos deberían diseñarse para reducirlo al mínimo. Los diseñadores del IPv6 entendieron muy bien este principio.

Una manera tentadora de acelerar el procedimiento es construir interfaces de red rápidas en hardware. La dificultad de esta estrategia es que a menos que el protocolo sea excesivamente sencillo, "hardware" simplemente significa una tarjeta de expansión con una segunda CPU y su propio programa. Para asegurar que el coprocesador de la red sea más económico que la CPU principal, con frecuencia se usa un chip más lento. La consecuencia de este diseño es que una buena parte del tiempo la CPU principal (rápida) tiene que esperar a que la segunda CPU (lenta) haga el trabajo crítico. Es un mito pensar que la CPU principal tiene otras tareas que hacer mientras espera. Es más, cuando dos CPU de propósito general se comunican, pueden ocurrir condiciones de competencia, por lo que se requieren protocolos complejos entre los dos procesadores para sincronizarlos correctamente y evitar estas condiciones. Por lo general, la mejor metodología es hacer que los protocolos sean sencillos y dejar que la CPU principal realice el trabajo.

La distribución de los paquetes es una consideración importante en las redes de gigabit. El encabezado debe contener la menor cantidad posible de campos para reducir el tiempo de procesamiento; además estos campos deben ser lo bastante grandes como para realizar su trabajo, y deben estar alineados por palabras para un procesamiento rápido. En este contexto, "lo bastante grandes" significa que no ocurran problemas como los números de secuencia que dan la vuelta mientras aún existen paquetes viejos, los receptores incapaces de anunciar suficiente espacio de ventana debido a que el campo de ventana es muy pequeño, etcétera.

El tamaño máximo de los datos debe ser un valor grande, para reducir la sobrecarga de software y permitir una operación eficiente. 1500 bytes es muy poco para las redes de alta velocidad, razón por la cual la Ethernet gigabit soporta tramas jumbo de hasta 9 KB, e IPv6 soporta paquetes de jumbogramas mayores de 64 KB.

Veamos ahora el asunto de la retroalimentación en los protocolos de alta velocidad. Debido al ciclo de retardo (relativamente) largo, hay que evitar la retroalimentación: la señalización del receptor al emisor tarda demasiado. Un ejemplo de retroalimentación es el control de la tasa de transmisión mediante un protocolo de ventana deslizante. Para evitar los retardos (largos) inherentes en el envío de actualizaciones de ventana del receptor al emisor, es mejor usar un protocolo basado en la tasa. En tal protocolo, el emisor puede enviar todo lo que quiera, siempre y cuando no envíe a mayor velocidad que cierta tasa acordada de antemano entre el emisor y el receptor.

Un segundo ejemplo de retroalimentación es el algoritmo de arranque lento de Jacobson. Este algoritmo efectúa múltiples sondeos para saber qué tanto puede manejar la red. Con las redes de alta velocidad, efectuar media docena de pequeños sondeos para ver la respuesta de la red es un desperdicio de ancho de banda enorme. Un esquema más eficiente es hacer que el emisor, el receptor y la red reserven los recursos necesarios al momento de establecer la conexión. Reservar los recursos por adelantado también tiene la ventaja de facilitar la reducción del jitter o variación del retardo. En pocas palabras, operar a altas velocidades inevitablemente empuja el diseño hacia la operación orientada a la conexión, o a algo muy parecido.

Otra valiosa característica es la habilidad de enviar una cantidad normal de datos junto con la solicitud de conexión. De esta forma podemos ahorrarnos un viaje de ida y vuelta.

6.7 REDES TOLERANTES AL RETARDO

Terminaremos este capítulo con la descripción de un nuevo tipo de transporte que tal vez algún día sea un componente importante de Internet. TCP y la mayoría de los otros protocolos de transporte se basan en la suposición de que el emisor y el receptor están conectados de manera continua mediante una trayectoria funcional, o de lo contrario el protocolo falla y no se pueden entregar los datos. En algunas redes es frecuente no contar con una trayectoria punto a punto. Un ejemplo de ello es la red espacial en la que los satélites LEO (Órbita Baja Terrestre) entran y salen del rango de las estaciones terrestres. Un satélite dado puede ser capaz de comunicarse con una estación terrestre sólo en tiempos específicos, y tal vez dos satélites nunca se puedan comunicar entre sí en ningún momento, incluso ni siquiera a través de una estación terrestre, porque tal vez uno de los satélites siempre esté fuera de rango. Otros ejemplos de redes involucran a los submarinos, autobuses, teléfonos móviles y otros dispositivos con computadoras, en los que hay una conectividad intermitente debido a la movilidad o las condiciones extremas.

En estas redes que se conectan en forma ocasional, para comunicar los datos se almacenan en nodos y se reenvían más tarde, cuando haya un enlace funcional. Esta técnica se denomina **conmutación de mensajes**. En un momento dado, los datos se transmitirán al destino. Una red cuya arquitectura se basa en esta metodología se llama **DTN** (**Red Tolerante al Retardo** o **Red Tolerante a las Interrupciones**, del inglés *Delay-Tolerant Network,* o *Disruption-Tolerant Network*).

El trabajo en las DTN empezó en 2002, cuando la IETF estableció un grupo de investigación sobre el tema. La inspiración de las DTN provino de una fuente insólita: los esfuerzos por enviar paquetes al espacio. Las redes espaciales deben lidiar con una comunicación intermitente y retardos muy largos. Kevin Fall observó que las ideas para estas interredes interplanetarias se podían aplicar a las redes en la Tierra, en donde la conectividad intermitente era la norma (Fall, 2003). Este modelo proporciona una útil generalización de Internet, en donde el almacenamiento y los retardos pueden ocurrir durante la comunicación. La entrega de datos es semejante a la entrega en el sistema postal, o el correo electrónico, en vez de la conmutación de paquetes en los enrutadores.

Desde 2002, la arquitectura de las DTN se ha refinado varias veces; además, las aplicaciones del modelo DTN han aumentado. Como aplicación dominante, considere los grandes conjuntos de datos de muchos terabytes que producen los experimentos científicos, los eventos de medios o los servicios basados en web, que necesitan copiarse a centros de datos en distintas ubicaciones en todo el mundo. A los operadores les gustaría enviar este tráfico en masa durante tiempos no pico para usar el ancho de banda que ya se pagó pero que no se utiliza, y están dispuestos a tolerar cierto retardo. Es como realizar las copias de seguridad en la noche, cuando no hay otras aplicaciones que usen mucho la red. El problema es que, para los servicios globales, los tiempos “no pico” son distintos en las diversas ubicaciones en todo el mundo. Puede haber un poco de traslape en los tiempos cuando los centros de datos en Boston y Perth tienen un ancho de banda de red no pico, debido a que la noche en una ciudad es el día en otra.

Sin embargo, los modelos DTN permiten el almacenamiento y los retardos durante la transferencia. Con este modelo, es posible enviar el conjunto de datos de Boston a Ámsterdam mediante el uso de ancho de banda no pico, puesto que las ciudades tienen zonas horarias con sólo seis horas de separación. A continuación, el conjunto de datos se almacena en Ámsterdam hasta que haya ancho de banda no pico entre Ámsterdam y Perth. Después se envía a Perth para completar la transferencia. Laoutaris y colaboradores (2009) estudiaron este modelo y descubrieron que puede proveer una capacidad considerable a muy bajo costo, además de que con frecuencia el uso de un modelo DTN duplica esa capacidad, si se le compara con un modelo tradicional punto a punto.

A continuación describiremos la arquitectura DTN de la IETF y los protocolos.

6.7.1 Arquitectura DTN

El principal supuesto en Internet que las DTN buscan relajar es que una trayectoria punto a punto entre un origen y un destino existe durante toda la sesión de comunicación. Cuando no se da este caso, los protocolos normales de Internet fallan. Las DTN resuelven la falta de conectividad punto a punto con una arquitectura basada en la conmutación de mensajes, como se muestra en la figura 6-56. También está diseñada para tolerar los enlaces con poca confiabilidad y retardos extensos. Esta arquitectura se especifica en el RFC 4838.

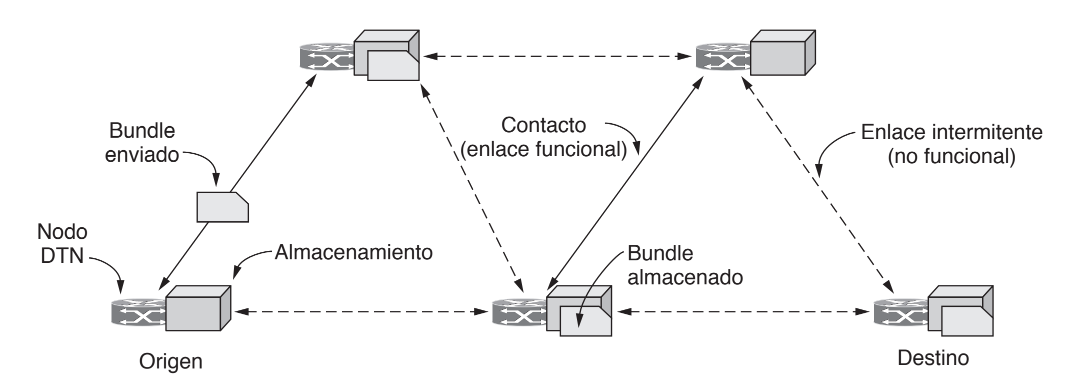


Figura 6-56. Arquitectura de red tolerante al retardo.

En terminología de DTN, a un mensaje se le llama **bundle**. Los nodos DTN están equipados con almacenamiento, que por lo general es persistente como un disco o memoria tipo Flash. Almacenan los bundles hasta que haya enlaces disponibles y después reenvían esos bundles. Los enlaces trabajan en forma intermitente. La figura 6-56 muestra cinco enlaces intermitentes que no están trabajando en ese momento, y dos enlaces que sí están trabajando. A un enlace funcional se le denomina **contacto**. La figura 6-56 también muestra unos bundles almacenados en dos nodos DTN que esperan a que los contactos envíen los bundles al siguiente punto en la trayectoria. De esta forma, los bundles se transmiten mediante los contactos, desde el origen hasta el destino.

El proceso de almacenar y reenviar los bundles en los nodos DTN es similar al proceso de encolar y reenviar los paquetes en los enrutadores, sólo que existen diferencias cualitativas. En los enrutadores de Internet, el encolamiento ocurre durante milisegundos, o cuando mucho, segundos. En los nodos DTN, los bundles pueden estar almacenados por horas hasta que llegue un autobús a la ciudad, mientras un avión completa su vuelo, hasta que un nodo sensor acumula suficiente energía solar como para funcionar, hasta que una computadora inactiva se despierta, etc. Estos ejemplos también indican una segunda diferencia: que los nodos se pueden desplazar (con un autobús o avión) mientras contienen datos almacenados, y este movimiento puede ser incluso una parte clave de la entrega de los datos. Los enrutadores en Internet no se pueden mover. El proceso en sí de mover los bundles se podría denominar mejor como “almacenamiento-transporte-reenvío”.

Como ejemplo, considere el escenario que se muestra en la figura 6-57, que fue el primer uso de los protocolos DTN en el espacio (Wood y colaboradores, 2008). El origen de los bundles es un satélite LEO que registra imágenes de la Tierra como parte de la Constelación de Satélites de Monitoreo de Desastres. Hay que devolver las imágenes al punto de recolección. Sin embargo, el satélite sólo tiene contacto intermitente con tres estaciones terrestres mientras orbita la Tierra. Entra en contacto con cada estación terrestre por turnos. El satélite, las estaciones terrestres y el punto de recolección actúan como un nodo DTN.

En cada contacto, se envía un bundle (o una porción del mismo) a una estación terrestre. A continuación, los bundles se envían a través de una red terrestre de soporte hasta el punto de recolección para completar la transferencia.

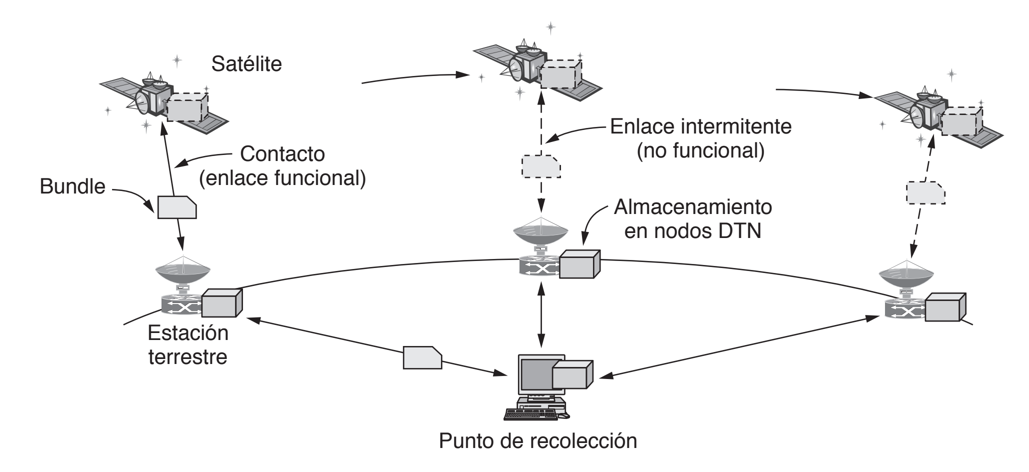


Figura 6-57. Uso de una red DTN en el espacio.

La principal ventaja de la arquitectura DTN en este ejemplo es que se adapta en forma natural a la situación en la que el satélite necesita almacenar imágenes, debido a que no hay conectividad en el momento en que se toma la imagen. También hay dos ventajas adicionales. En primer lugar, tal vez no haya un solo contacto que dure lo suficiente como para enviar las imágenes. Sin embargo, éstas se pueden dispersar por los contactos con tres estaciones terrestres. En segundo lugar, el uso del enlace entre el satélite y la estación terrestre está separado del enlace a través de la red de soporte. Esto significa que la descarga del satélite no está limitada por un enlace terrestre lento. Puede proceder a toda velocidad y el bundle se almacenará en la estación terrestre hasta que se pueda transmitir al punto de recolección.

Un aspecto importante que la arquitectura no especifica es cómo encontrar buenas rutas a través de nodos DTN. Una ruta a usar en esta trayectoria. Las buenas rutas dependen de la naturaleza de la arquitectura que describe cuándo enviar los datos, y también a qué contactos. Algunos de ellos se conocen de antemano. Un buen ejemplo es el movimiento de los cuerpos celestiales en el ejemplo del espacio. En el experimento espacial se sabía de antemano cuándo ocurrirían los contactos, que los intervalos de los contactos variaban de 5 a 14 minutos por pasada con cada estación terrestre, y que la capacidad del enlace descendente era de 8.134 Mbps. Con base en este conocimiento, el transporte de un bundle de imágenes se puede planear por adelantado.

En otros casos es posible predecir los contactos, sólo que con menos certeza. Como ejemplo tenemos a los autobuses que hacen contacto unos con otros de manera regular, gracias a un cronograma, aunque con algunas variaciones, en los tiempos y la cantidad de ancho de banda de poca actividad en las redes de ISP, que se predicen con base en los datos anteriores. En el otro extremo, los contactos son ocasionales y aleatorios. Un ejemplo es el transporte de datos de un usuario a otro en los teléfonos móviles, dependiendo de cuáles usuarios entren en contacto con otros durante el día. Cuando hay incertidumbre en los contactos, una estrategia de enrutamiento es enviar copias del bundle a lo largo de distintas trayectorias con la esperanza de que una de las copias se entregue al destino antes de que se agote el tiempo de vida.

6.7.2 El protocolo Bundle

Para ver con más detalle la operación de las redes DTN, vamos a analizar los protocolos de la IETF. Las redes DTN son un tipo de red emergente; las DTN experimentales han usado distintos protocolos, puesto que no hay un requerimiento para usar los protocolos de la IETF. Sin embargo, por lo menos son un buen punto para empezar y resaltar muchas de las cuestiones clave.

La pila de protocolos DTN se muestra en la figura 6-58. El protocolo clave es el **protocolo Bundle**, que se especifica en el RFC 5050. Este protocolo es responsable de aceptar los mensajes de la aplicación y enviarlos como uno o más bundles por medio de operaciones de almacenamiento-transporte-reenvío hacia el nodo DTN de destino. En la figura 6-58 también podemos ver que el protocolo Bundle opera encima del nivel de TCP/IP. En otras palabras, podemos usar TCP/IP sobre cada contacto para mover los bundles entre un nodo DTN y otro. Este posicionamiento trae a relucir la cuestión de si el protocolo Bundle es un protocolo de capa de transporte o de capa de aplicación. Al igual que con el RTP, nuestra opinión es que, a pesar de operar sobre un protocolo de transporte, el protocolo Bundle provee un servicio de transporte a muchas aplicaciones diferentes, por lo que cubriremos las redes DTN en este capítulo.

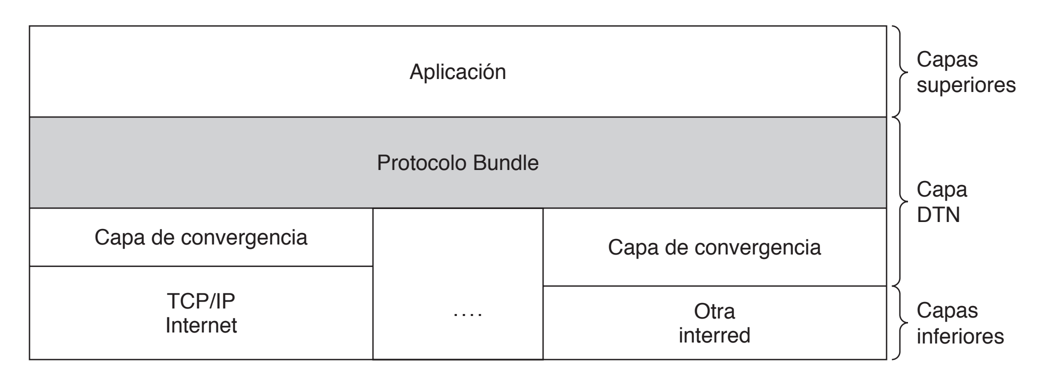


Figura 6-58. Pila de protocolos de redes tolerantes al retardo.

En la figura 6-58 podemos ver que el protocolo Bundle puede operar sobre otros tipos de protocolos, como UDP, o incluso sobre otros tipos de interredes. Por ejemplo, en una red espacial los enlaces pueden tener retardos muy largos. El tiempo de ida y vuelta entre la Tierra y Marte puede ser de hasta 20 minutos, dependiendo de la posición relativa de los planetas. Imagine cómo funcionarán las confirmaciones de recepción y las retransmisiones TCP en ese enlace, sobre todo para los mensajes relativamente cortos. Nada bien, por supuesto. En cambio, podríamos usar otro protocolo que utilice códigos de corrección de errores. O en las redes de sensores que tienen sus recursos muy restringidos podríamos usar un protocolo más ligero que TCP.

Como el protocolo Bundle es fijo, aunque está diseñado para operar sobre una variedad de transportes, debe haber un vacío en la funcionalidad entre los protocolos. Ese vacío es la razón por la que se incluye una capa de convergencia en la figura 6-58. Esta capa de convergencia es simplemente una capa de pegamento que asocia las interfaces de los protocolos que une. Por definición, hay una capa de convergencia diferente para cada una de las distintas capas de transporte inferiores. Las capas de convergencia se encuentran comúnmente en los estándares para unirse a protocolos nuevos y existentes.

En la figura 6-59 se muestra el formato de los mensajes del protocolo Bundle. Los distintos campos en estos mensajes nos indican algunas de las cuestiones clave que maneja el protocolo Bundle.

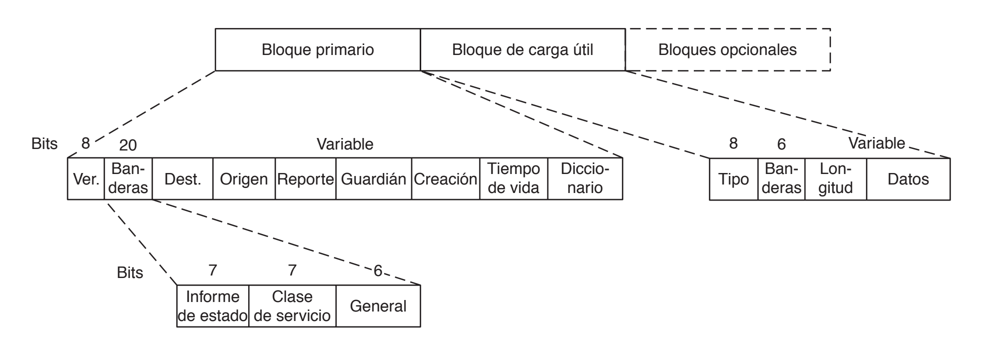


Figura 6-59. Formato de mensajes del protocolo Bundle.

Cada mensaje consiste en un bloque primario, que se puede considerar como un encabezado, un bloque de carga útil para los datos y otros bloques opcionales; por ejemplo, para transportar parámetros de seguridad. El bloque primario empieza con un campo *Versión* (en la actualidad es la 6) seguido de un campo *Banderas*. Entre otras funciones, las banderas codifican una clase de servicio para permitir que un origen marque sus bundles como de alta o baja prioridad, además de otras solicitudes de manejo; por ejemplo, si el destino debe confirmar la recepción del bundle.

Después vienen las direcciones, que resaltan tres partes interesantes del diseño. Además de los campos identificadores *Destino* y *Origen*, hay un campo identificador *Guardián*. El guardián es la parte responsable de ver que se entregue el bundle. En Internet, por lo general el nodo de origen es el guardián, ya que es el nodo que retransmite los datos si no se entregan en última instancia al destino. No obstante, en una DTN el nodo de origen no siempre puede estar conectado, por lo que tal vez no tenga forma de saber si se entregaron o no los datos. Para lidiar con este problema, las redes DTN usan la noción de **transferencia de guardián**, en donde otro nodo cercano al destino puede asumir la responsabilidad de ver que se entreguen los datos en forma segura. Por ejemplo, si se almacena un bundle en un avión para reenviarlo más tarde y en otra ubicación, el avión se puede convertir en el guardián del bundle.

El segundo aspecto interesante es que estos identificadores *no* son direcciones IP. Como el protocolo bundle está diseñado para funcionar a través de una variedad de transportes e interredes, define sus propios identificadores. En realidad estos identificadores son más como nombres de alto nivel (como los URL de las páginas web) que direcciones de bajo nivel (como las direcciones IP). Proporcionan a las redes DTN un aspecto de enrutamiento a nivel de aplicación, como la entrega de correo electrónico o la distribución de actualizaciones de software.

El tercer aspecto interesante es la forma en que se codifican los identificadores. También existe un identificador *Informe* para los mensajes de diagnóstico. Todos los identificadores se codifican como referencias a un campo *Diccionario* de longitud variable. Este campo provee compresión cuando los nodos guardián o de informe son iguales que el origen o el destino. De hecho, la mayor parte del formato del mensaje se diseñó teniendo en mente tanto la extensibilidad como la eficiencia, mediante el uso de una representación compacta de campos de longitud variable. La representación compacta es importante para los enlaces inalámbricos y los nodos con recursos limitados, como en una red de sensores.

A continuación viene un campo *Creación*, el cual transporta la hora en que se creó el bundle, junto con un número de secuencia proveniente del origen para mantener el orden, además de un campo *Tiempo de vida* que indica el lapso después del cual los datos del bundle ya no serán de utilidad. Estos campos existen debido a que los datos se pueden almacenar durante un largo periodo en los nodos DTN, por lo que debe haber alguna forma de quitar los datos viejos de la red. A diferencia de Internet, requieren que los nodos DTN tengan relojes con una sincronización con poca exactitud.

El bloque primario se completa con el campo *Diccionario*. Después viene el bloque de carga útil. Este bloque empieza con un campo *Tipo* corto que lo identifica como carga útil, seguido de un pequeño conjunto de *Banderas* que describen las opciones de procesamiento. Luego viene el campo *Datos*, precedido de un campo *Longitud*. Por último, puede haber otros bloques opcionales, como un bloque que transporta parámetros de seguridad.

Hay muchos aspectos de las redes DTN que se están explorando en la comunidad de investigación. Las buenas estrategias de enrutamiento dependen de la naturaleza de los contactos, como se mencionó antes. Al almacenar los datos dentro de la red se generan otros problemas. Ahora el control de la congestión debe considerar el almacenamiento en los nodos como otro tipo de recurso que se puede agotar. La falta de comunicación punto a punto también exacerba los problemas de seguridad. Antes de que un nodo DTN se convierta en el guardián de un bundle, tal vez sea conveniente que sepa que el emisor está autorizado a usar la red y que quizás el destino esté buscando ese bundle. Las soluciones a estos problemas dependerán del tipo de DTN, puesto que las redes espaciales son distintas de las redes de sensores.

6.8 RESUMEN

La capa de transporte es la clave para entender los protocolos en capas. Esta capa proporciona varios servicios, siendo el más importante un flujo de bytes punto a punto, confiable y orientado a conexión desde el emisor hasta el receptor. Para acceder a esta capa se utilizan primitivas de servicio que permiten establecer, usar y liberar conexiones. Una interfaz común de la capa de transporte es la que proporcionan los sockets de Berkeley.

Los protocolos de transporte deben ser capaces de administrar conexiones a través de redes no confiables. El establecimiento de las conexiones se complica por la existencia de paquetes duplicados retardados que pueden reaparecer en momentos inoportunos. Para lidiar con ellos, se requieren acuerdos de tres vías para establecer las conexiones. Es más fácil liberar una conexión que establecerla, pero aun así este proceso está lejos de ser trivial debido al problema de los dos ejércitos.

Incluso cuando la capa de red es completamente confiable, la capa de transporte tiene mucho trabajo por hacer. Debe manejar todas las primitivas de servicio, administrar conexiones y temporizadores, asignar ancho de banda con el control de congestión, y ejecutar una ventana deslizante de tamaño variable para el control de flujo.

El control de la congestión debe asignar en forma equitativa todo el ancho de banda disponible entre los flujos que compiten entre sí; además debe rastrear los cambios en el uso de la red. La ley de control AIMD converge a una asignación justa y equitativa.

Internet tiene dos protocolos de transporte principales: UDP y TCP. UDP es un protocolo sin conexión que actúa principalmente como una envoltura para los paquetes IP con la característica adicional de multiplexar y desmultiplexar diversos procesos mediante el uso de una sola dirección IP. UDP se puede emplear para interacciones cliente-servidor; por ejemplo, mediante el uso de RPC. También se puede emplear para construir protocolos en tiempo real tales como RTP.

El protocolo de transporte principal de Internet es TCP. Proporciona un flujo de bytes confiable, bidireccional y controlado por congestión, con un encabezado de 20 bytes en todos los segmentos. Se ha invertido mucho trabajo en optimizar el desempeño de TCP, mediante los algoritmos de Nagle, Clark, Jacobson, Karn, entre otros.

Por lo general, el desempeño de las redes es dominado por la sobrecarga de procesamiento de los protocolos y los segmentos; y esta situación empeora a mayores velocidades. Los protocolos deberían diseñarse para reducir al mínimo la cantidad de segmentos y el trabajo para las trayectorias con un producto ancho de banda-retardo grande. En las redes de gigabits se requieren protocolos sencillos y un procesamiento modernizado.

Las redes tolerantes al retardo ofrecen un servicio de entrega a través de redes con una conectividad ocasional o retardos largos entre los enlaces. Los nodos intermedios almacenan, transportan y reenvían bundles de información para que se entregue en un momento dado, incluso aunque no haya una trayectoria funcional del emisor al receptor en ningún momento.

PROBLEMAS

1. En nuestras primitivas de transporte de ejemplo de la figura 6-2, LISTEN es una llamada bloqueadora. ¿Es estrictamente necesario esto? De no serlo, explique cómo debe usarse una primitiva no bloqueadora. ¿Qué ventaja tendría esto respecto al esquema descrito en el texto?

2. Las primitivas del servicio de transporte asumen una asimetría entre los dos puntos terminales durante el establecimiento de una conexión. Un punto (servidor) ejecuta LISTEN mientras que el otro (cliente) ejecuta CONNECT. Sin embargo, en las aplicaciones de igual a igual como los sistemas para compartir archivos (por ejemplo, BitTorrent), todos los puntos terminales son iguales. No hay servidor ni funcionalidad de cliente. ¿Cómo se pueden usar las primitivas de servicio de transporte para crear tales aplicaciones de igual a igual?

3. En el modelo subyacente de la figura 6-4, se supone que la capa de red puede perder paquetes y, por tanto, su recepción se debe confirmar por separado. Suponga que la capa de red es 100% confiable y que nunca pierde paquetes. ¿Qué cambios, si acaso, se necesitarán en la figura 6-4?

4. En las dos partes de la figura 6-6 hay un comentario que indica que el valor de SERVER_PORT debe ser igual tanto en el cliente como en el servidor. ¿Por qué es esto tan importante?

5. En el ejemplo del servidor de archivos de Internet (figura 6-6), ¿puede fallar la llamada de sistema connect( ) por alguna razón que no sea que la cola de escucha esté llena en el servidor? Suponga que la red es perfecta.

6. Un criterio para decidir entre tener un servidor activo todo el tiempo o hacer que inicie bajo demanda mediante un servidor de procesos es la frecuencia con que se utiliza el servicio proporcionado. ¿Puede usted pensar en algún otro criterio para tomar esta decisión?

7. Suponga que el esquema controlado por reloj para generar números de secuencia iniciales se utiliza con un contador de reloj de 15 bits de anchura. El reloj pulsa una vez cada 100 mseg, y el tiempo de vida máximo de paquete es de 60 segundos. ¿Con qué frecuencia debe ocurrir una resincronización:
(a) en el peor caso?
(b) cuando los datos consumen 240 números de secuencia/minuto?

8. ¿Por qué tiene que ser el tiempo de vida máximo de paquete, *T*, lo bastante grande para asegurar que no sólo haya desaparecido el paquete, sino también sus confirmaciones de recepción?

9. Imagine que se usa un acuerdo de dos vías en lugar de uno de tres vías para establecer las conexiones. En otras palabras, no se requiere el tercer mensaje. ¿Son posibles ahora los interbloqueos? Dé un ejemplo o demuestre que no pueden existir.

10. Imagine un problema de *n* ejércitos generalizado, en el que el acuerdo de dos de los ejércitos azules sea suficiente para la victoria. ¿Existe un protocolo que permita ganar a los azules?

11. Considere el problema de recuperarse de las fallas de host (como en la figura 6-18). Si el intervalo entre escribir y enviar una confirmación de recepción (o viceversa) se puede hacer relativamente pequeño, ¿cuáles son las dos mejores estrategias de emisor-receptor para minimizar la probabilidad de que el protocolo falle?

12. En la figura 6-20, suponga que se agrega un nuevo flujo *E* que toma una trayectoria de *R1* a *R2* a *R6*. ¿Cómo cambia la asignación de ancho de banda con equidad máxima-mínima para los cinco flujos?

13. Explique las ventajas y desventajas de los créditos en comparación con los protocolos de ventana deslizante.

14. Algunas otras políticas para la equidad en el control de la congestión son el Incremento Aditivo/Decremento Aditivo (AIAD), Incremento Multiplicativo/Decremento Aditivo (MIAD) e Incremento Multiplicativo/Decremento Multiplicativo (MIMD). Explique estas tres políticas en términos de convergencia y estabilidad.

15. ¿Por qué existe el UDP? ¿No hubiera bastado con permitir que los procesos de usuario enviaran paquetes IP puros?

16. Considere un protocolo de nivel de aplicación simple construido encima de UDP, que permite a un cliente recuperar un archivo desde un servidor remoto que reside en una dirección bien conocida. El cliente primero envía una solicitud con el nombre del archivo, y el servidor responde con una secuencia de paquetes de datos que contienen diferentes partes del archivo solicitado. Para asegurar la confiabilidad y una entrega en secuencia, el cliente y el servidor utilizan un protocolo de parada y espera. Si ignoramos el aspecto de desempeño obvio, ¿ve usted un problema con este protocolo? Piense con cuidado en la posibilidad de que fallen los procesos.

17. Un cliente envía una solicitud de 128 bytes a un servidor localizado a 100 km de distancia a través de una fibra óptica de 1 gigabit. ¿Cuál es la eficiencia de la línea durante la llamada al procedimiento remoto?

18. Considere de nuevo la situación del problema anterior. Calcule el tiempo de respuesta mínimo posible para la línea de 1 Gbps y para una de 1 Mbps. ¿Qué conclusión puede obtener?

19. Tanto UDP como TCP utilizan números de puerto para identificar la entidad de destino al entregar un mensaje. Dé dos razones por las cuales estos protocolos inventaron un nuevo ID abstracto (números de puerto), en lugar de utilizar el ID del proceso, que ya existían cuando se diseñaron estos protocolos.

20. Varias implementaciones de RPC proveen una opción para que el cliente use RPC implementada sobre UDP, o RPC implementada sobre TCP. ¿Bajo qué condiciones preferirá un cliente usar RPC sobre UDP, y bajo qué condiciones preferirá usar RPC sobre TCP?

21. Considere dos redes, *N1* y *N2*, que tienen el mismo retardo promedio entre un origen *A* y un destino *D*. En *N1*, el retardo experimentado por los distintos paquetes se distribuye de manera uniforme con un retardo máximo de 10 segundos, mientras que en *N2*, 99% de los paquetes experimentan un retardo de menos de un segundo sin límite en el retardo máximo. Explique cómo se puede usar RTP en estos casos para transmitir el flujo de audio/video en vivo.

22. ¿Cuál es el tamaño total de la MTU mínima de TCP, incluyendo la sobrecarga de TCP e IP pero no la sobrecarga de la capa de enlace de datos?

23. La fragmentación y el reensamble de datagramas se manejan a través del IP y son transparentes para TCP. ¿Significa esto que TCP no tiene que preocuparse porque los datos lleguen en el orden equivocado?

24. RTP se utiliza para transmitir audio con calidad de CD, el cual crea un par de muestras de 16 bits 44,100 veces/seg, una muestra por cada uno de los canales de estéreo. ¿Cuántos paquetes por segundo debe transmitir RTP?

25. ¿Sería posible colocar el código RTP en el kernel del sistema operativo, junto con el código UDP? Explique su respuesta.

26. Un proceso en el host 1 se ha asignado al puerto *p*, y un proceso en el host 2 se ha asignado al puerto *q*. ¿Es posible que haya dos o más conexiones TCP entre estos dos puertos al mismo tiempo?

27. En la figura 6-36 vimos que además del campo *Confirmación de recepción* de 32 bits, hay un bit *ACK* en la cuarta palabra. ¿Acaso esto agrega realmente algo? ¿Por qué sí o por qué no?

28. La máxima carga útil de un segmento TCP es de 65,495 bytes. ¿Por qué se eligió ese extraño número?

29. Describa dos formas de entrar al estado *SYN RCVD* de la figura 6-39.

30. Considere el efecto de usar arranque lento en una línea con un tiempo de ida y vuelta de 10 mseg sin congestión. La ventana receptora es de 24 KB y el tamaño máximo de segmento es de 2 KB. ¿Cuánto tiempo pasará antes de enviar la primera ventana completa?

31. Suponga que la ventana de congestión del TCP está ajustada a 18 KB y que expira el temporizador. ¿Qué tan grande será la ventana si las siguientes cuatro ráfagas de transmisiones tienen éxito? Suponga que el tamaño máximo de segmento es de 1 KB.

32. Si el tiempo de ida y vuelta actual del TCP (*RTT*) es de 30 mseg y las siguientes confirmaciones de recepción llegan después de 26, 32 y 24 mseg, respectivamente, ¿cuál es la nueva estimación de *RTT* si se utiliza el algoritmo de Jacobson? Use $\alpha = 0.9$.

33. Una máquina TCP envía ventanas completas de 65 535 bytes por un canal de 1 Gbps que tiene un retardo de 10 mseg en un solo sentido. ¿Cuál es la tasa máxima de transferencia real que se puede lograr? ¿Cuál es la eficiencia de la línea?

34. ¿Cuál es la velocidad máxima de línea a la que un host puede enviar cargas útiles TCP de 1500 bytes con un tiempo de vida máximo de paquete de 120 seg, sin que los números de secuencia den vuelta? Tome en cuenta la sobrecarga TCP, IP y Ethernet. Suponga que las tramas Ethernet se pueden enviar de manera continua.

35. Para lidiar con las limitaciones de la versión 4 del IP, hubo que realizar un importante esfuerzo por medio de la IETF, que resultó en el diseño de la versión 6 del IP pero aún hay mucha resistencia en cuanto a la adopción de esta nueva versión. Sin embargo, no se requiere ningún esfuerzo importante para lidiar con las limitaciones de TCP. Explique por qué.

36. En una red cuyo segmento máximo es de 128 bytes, el tiempo de vida máximo del segmento es de 30 segundos y tiene números de secuencia de 8 bits, ¿cuál es la tasa máxima de datos por conexión?

37. Suponga que usted está midiendo el tiempo para recibir un segmento. Cuando ocurre una interrupción, lee el reloj del sistema en milisegundos. Cuando el segmento se procesa por completo, lee nuevamente el reloj. Mide 0 mseg 270 000 veces y 1 mseg 730 000 veces. ¿Cuánto tiempo tarda en recibir un segmento?

38. Una CPU ejecuta instrucciones a la tasa de 1000 MIPS. Los datos se pueden copiar 64 bits a la vez, y cada palabra requiere 10 instrucciones para copiarse. Si un paquete entrante tiene que copiarse cuatro veces, ¿puede este sistema manejar una línea de 1 Gbps? Por simplicidad, suponga que todas las instrucciones, incluso aquellas que leen o escriben en la memoria, se ejecutan a la tasa total de 1000 MIPS.

39. Para resolver el problema de los números de secuencia que dan la vuelta mientras los paquetes anteriores aún existen, podríamos utilizar números de secuencia de 64 bits. Sin embargo, en teoría una fibra óptica puede operar a 75 Tbps. ¿Cuál es el tiempo de vida máximo de paquete que se requiere para asegurarse de que las futuras redes de 75 Tbps no tengan problemas de números de secuencia que den vuelta, incluso con los números de secuencia de 64 bits? Suponga que cada byte tiene su propio número de secuencia, al igual que TCP.

40. En la sección 6.6.5 calculamos que una línea de gigabits descarga 80 000 paquetes/seg en el host, y éste dispone de sólo 6 250 instrucciones para procesarlos, lo que deja la mitad del tiempo de CPU para las aplicaciones. Para este cálculo se asumió un tamaño de paquete de 1500 bytes. Rehaga el cálculo para un paquete de tamaño de ARPANET (128 bytes). En ambos casos, suponga que los tamaños de paquete dados incluyen toda la sobrecarga.

41. Para una red de 1 Gbps que opera sobre 4 000 km, el retardo es el factor limitante, no el ancho de banda. Considere una MAN con un promedio de 20 km de distancia entre el origen y el destino. ¿A qué tasa de datos el retardo de ida y vuelta debido a la velocidad de la luz iguala el retardo de transmisión para un paquete de 1 KB?

42. Calcule el producto ancho de banda-retardo para las siguientes redes: (1) T1 (1.5 Mbps), (2) Ethernet (10 Mbps); (3) T3 (45 Mbps) y (4) STS-3 (155 Mbps). Suponga un RTT de 100 mseg. Recuerde que un encabezado TCP tiene 16 bits reservados para el tamaño de ventana. ¿Cuáles son sus implicaciones a la luz de sus cálculos?

43. ¿Cuál es el producto ancho de banda-retardo para un canal de 50 Mbps en un satélite geoestacionario? Si todos los paquetes son de 1500 bytes (incluyendo la sobrecarga), ¿qué tan grande debería ser la ventana en los paquetes?

44. El servidor de archivos de la figura 6-6 está muy lejos de ser perfecto y le vendrían bien algunas mejoras. Realice las siguientes modificaciones.
(a) Dé al cliente un tercer argumento que especifique un rango de bytes.
(b) Agregue una bandera –w al cliente, que permita escribir el archivo en el servidor.

45. Una función común que necesitan todos los protocolos es la manipulación de mensajes. Recuerde que los protocolos manipulan mensajes mediante la adición/eliminación de encabezados. Algunos protocolos pueden dividir un solo mensaje en varios fragmentos, para después unir estos fragmentos de vuelta en un solo mensaje. Para este fin, diseñe e implemente una biblioteca de administración de mensajes que ofrezca soporte para crear un nuevo mensaje, adjuntar un encabezado a un mensaje, eliminar un encabezado de un mensaje, descomponer un mensaje en dos mensajes, combinar dos mensajes en un solo mensaje y guardar una copia de un mensaje. Su implementación debe minimizar la copia de datos de un búfer a otro lo más que sea posible. Es imprescindible que las operaciones que manipulan mensajes no toquen los datos en un mensaje, sino que sólo manipulen apuntadores.

46. Diseñe e implemente un sistema de salón de conversación (chat) que permita conversar a varios grupos de usuarios. Un coordinador del salón reside en una dirección bien conocida de red, utiliza UDP para comunicarse con los clientes del salón, configura los servidores del salón para cada sesión de conversación y mantiene un directorio de sesiones de conversación. Hay un servidor de conversación por sesión. Un servidor de este tipo utiliza TCP para comunicarse con los clientes. Un cliente de conversación permite que los usuarios inicien, se unan y dejen una sesión de conversación. Diseñe e implemente el código del coordinador, del servidor y del cliente.

7

LA CAPA DE APLICACIÓN

Habiendo concluido todos los preliminares, ahora llegamos a la capa en donde se encuentran todas las aplicaciones. Las capas por debajo de la de aplicación están ahí para proporcionar servicios de transporte, pero no hacen ningún trabajo verdadero para los usuarios. En este capítulo estudiaremos algunas aplicaciones de red reales.

Sin embargo, incluso en la capa de aplicación se necesitan protocolos de apoyo que permitan el funcionamiento de las aplicaciones. En consecuencia, veremos uno de estos protocolos importantes antes de comenzar con las aplicaciones. El elemento en cuestión es el DNS, que maneja la asociación de nombres dentro de Internet. Después examinaremos tres aplicaciones reales: correo electrónico, World Wide Web y multimedia. Para terminar el capítulo veremos más detalles sobre la distribución de contenido, incluyendo la distribución mediante las redes de igual a igual.

7.1 DNS: EL SISTEMA DE NOMBRES DE DOMINIO

Aunque en teoría los programas pueden hacer referencia a páginas web, buzones de correo y otros recursos mediante las direcciones de red (por ejemplo, IP) de las computadoras en las que se almacenan, a las personas se les dificulta recordar estas direcciones. Además, navegar en las páginas web de una empresa desde un servidor con la dirección *128.111.24.41* significa que si la compañía mueve el servidor web a una máquina diferente con una dirección IP distinta, hay que avisar a todos sobre la nueva dirección IP. Por este motivo se introdujeron nombres legibles de alto nivel con el fin de separar los nombres de máquina de las direcciones de máquina. De esta manera, el servidor web de la empresa podría conocerse como *www.cs.washington.edu* sin importar cuál sea su dirección IP. Sin embargo, como la red sólo comprende direcciones numéricas, se requiere algún mecanismo para convertir los nombres en direcciones de red. En las siguientes secciones analizaremos la forma en que se logra esta correspondencia.

Hace mucho, en los tiempos de ARPANET sólo había un archivo, *hosts.txt*, en el que se listaban todos los nombres de computadoras y sus direcciones IP. Cada noche, todos los hosts obtenían este archivo del sitio en el que se almacenaba. En una red conformada por unos cuantos cientos de grandes máquinas de tiempo compartido este método funcionaba razonablemente bien.

Sin embargo, antes de que se conectaran millones de computadoras PC a Internet, los involucrados se dieron cuenta de que este método no funcionaría eternamente. Por una parte, el tamaño de cada archivo crecería de manera considerable. Sin embargo, un problema aún mayor eran los constantes conflictos con los nombres de los hosts, a menos que dichos nombres se administraran en forma central (algo impensable en una enorme red internacional debido a su carga y latencia). Para resolver estos problemas, en 1983 se inventó el **DNS** (**Sistema de Nombres de Dominio**, del inglés *Domain Name System*), el cual ha sido una parte clave de Internet desde entonces.

La esencia del DNS es la invención de un esquema jerárquico de nombres basado en dominios y un sistema de base de datos distribuido para implementar este esquema de nombres. El DNS se usa principalmente para asociar los nombres de host con las direcciones IP, pero también se puede usar para otros fines. El DNS se define en los RFC 1034, 1035, 2181 y se elabora con más detalle en muchos otros.

Dicho en forma muy breve, la forma en que se utiliza el DNS es la siguiente. Para asociar un nombre con una dirección IP, un programa de aplicación llama a un procedimiento de biblioteca llamado **resolvedor** y le pasa el nombre como parámetro. En la figura 6.6 vimos un ejemplo de un resolvedor, *gethostbyname*. El cual envía una consulta que contiene el nombre a un servidor DNS local, que después busca el nombre y devuelve una respuesta con la dirección IP al resolvedor, que a su vez lo devuelve al solicitante. Los mensajes de solicitud y respuesta se envían como paquetes UDP. Armado con la dirección IP, el programa puede entonces establecer una conexión TCP con el host o enviarle paquetes UDP.

7.1.1 El espacio de nombres del DNS

La administración de un conjunto grande de nombres que cambian en forma continua no es un problema sencillo. En el sistema postal, para administrar los nombres se requieren letras que especifiquen (de manera implícita o explícita) el país, estado o provincia, ciudad, calle y nombre del destinatario. Con este tipo de direccionamiento jerárquico, se asegura que no haya confusión entre el Marvin Anderson, de Main St., en White Plains, N.Y., y el Marvin Anderson, de Main St., en Austin, Texas. El DNS funciona de la misma manera.

En Internet, el nivel superior de la jerarquía de nombres se administra mediante una organización llamada **ICANN** (**Corporación de Internet para la Asignación de Nombres y Números**, del inglés *Internet Corporation for Assigned Names and Numbers*), la cual se creó para este fin en 1998 como parte del proceso de maduración de Internet, que se convirtió en un asunto económico a nivel mundial. En concepto, Internet se divide en más de 250 **dominios de nivel superior**, cada uno de los cuales abarca muchos hosts. Cada dominio se divide en subdominios, los que a su vez también se dividen, y así en lo sucesivo. Todos estos dominios se pueden representar mediante un árbol, como se muestra en la figura 7-1. Las hojas del árbol representan los dominios que no tienen subdominios (pero que, por supuesto, contienen máquinas). Un dominio de hoja puede contener un solo host, o puede representar a una compañía y contener miles de hosts.

Los dominios de nivel superior se dividen en dos categorías: genéricos y países. Los dominios genéricos, que se listan en la figura 7-2, incluyen los dominios originales de la década de 1980 y los dominios introducidos mediante solicitudes a la ICANN. En lo futuro se agregarán otros dominios genéricos de nivel superior.

Los dominios de país incluyen una entrada para cada país, como se define en la ISO 3166. En 2010 se introdujeron nombres de dominio internacionalizados de países en los que se utilizan alfabetos distintos al latín. Estos dominios permiten nombrar hosts en árabe, cirílico, chino u otros idiomas.

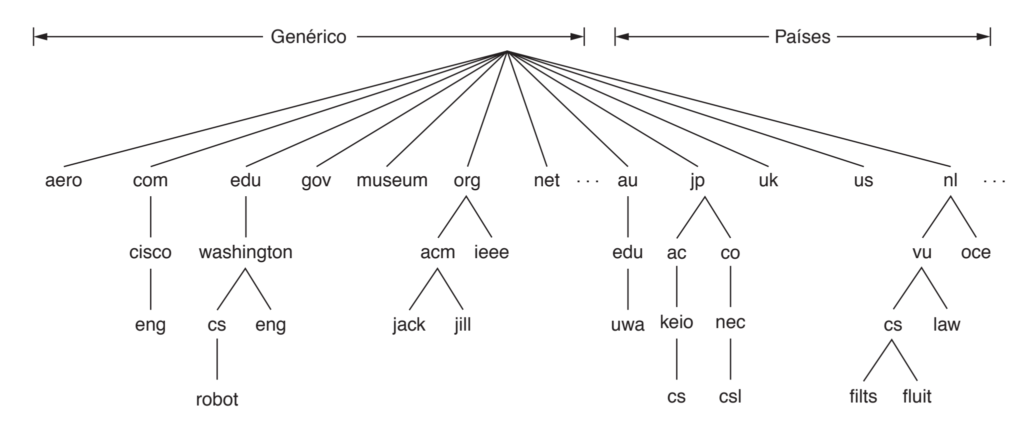


Figura 7-1. Una porción del espacio de nombres de dominio de Internet.

En general, es fácil obtener un dominio de segundo nivel, como *nombre-de-compañía.com*. Los dominios de nivel superior son operados por **registradores** nombrados por la ICANN. Sólo es necesario ir con el registrador correspondiente (*com*, en este caso) para ver si el nombre deseado está disponible y si no es la marca registrada de alguien más. Si no hay problemas, el solicitante paga una pequeña cuota anual y obtiene el nombre.

Sin embargo, a medida que Internet se ha vuelto más comercial e internacional, también ha generado más polémica, en especial en cuestiones relacionadas con los nombres. Esta controversia incluye a la misma ICANN. Por ejemplo, la creación del dominio *xxx* tardó varios años y juicios para resolverse. ¿Es bueno o malo colocar de manera voluntaria el contenido para adultos en su propio dominio? (Algunas personas no querían ningún contenido para adultos disponible en Internet, mientras que otros querían ponerlo todo en un solo dominio, de modo que los filtros de protección pudieran encontrar esos sitios fácilmente y apartarlos de los niños). Algunos de los dominios se organizan de manera automática, mientras que otros tienen restricciones en cuanto a quién pueden obtener un nombre, como se observa en la figura 7-2. Pero, ¿qué restricciones son apropiadas? Tomemos como ejemplo el dominio *pro*. Es para profesionales calificados; pero, ¿a quién se puede designar profesional? Es evidente que los doctores y abogados son profesionales. Y, ¿qué hay sobre los fotógrafos independientes, profesores de piano, magos, plomeros, peluqueros, exterminadores, tatuadores, mercenarios y prostitutas? ¿Son elegibles estas ocupaciones? ¿De acuerdo con quién?

También hay dinero en los nombres. Tuvalu (el país) otorgó un contrato de arrendamiento de su domino *tv* por $50 millones, todo porque el código del país se adapta bien para promocionar los sitios de televisión. Casi cualquier palabra común (en inglés) está ocupada en el dominio *com*, además de los errores ortográficos más comunes. Intente con artículos para el hogar, animales, plantas, partes del cuerpo, etc. La práctica de registrar un dominio con miras a venderlo después a una parte interesada a un precio mucho mayor tiene incluso su propio nombre: **ciberocupación** (*cybersquatting*). Muchas empresas que tuvieron un arranque lento cuando empezó la era de Internet, descubrieron que sus nombres de dominio ya estaban ocupados para cuando trataron de adquirirlos. En general, mientras no se violen marcas registradas y no haya fraude, el primero que llega es el primero que obtiene el nombre. Sin embargo, aún se están refinando políticas para resolver las disputas de nombres.

Cada dominio se nombra por la ruta hacia arriba desde él a la raíz (sin nombre). Los componentes se separan mediante puntos. Así, el departamento de ingeniería en Cisco podría ser *eng.cisco.com.*, en lugar de un nombre al estilo UNIX como */com/cisco/eng*. Observe que esta asignación jerárquica significa que *eng.cisco.com.* no tiene conflicto con el uso potencial de *eng* en *eng.washington.edu.*, que podría usarse en el departamento de inglés de la Universidad de Washington.

Dominio	Uso deseado	Fecha de inicio	¿Restringido?
com	Comercial	1985	No
edu	Instituciones educativas	1985	Sí
gov	Gobierno	1985	Sí
int	Organizaciones internacionales	1988	Sí
mil	Milicia	1985	Sí
net	Proveedores de red	1985	No
org	Organizaciones sin fines de lucro	1985	No
aero	Transporte aéreo	2001	Sí
biz	Negocios	2001	No
coop	Cooperativas	2001	Sí
info	Informacional	2002	No
museum	Museos	2002	Sí
name	Personas	2002	No
pro	Profesionales	2002	Sí
cat	Catalán	2005	Sí
jobs	Empleo	2005	Sí
mobi	Dispositivos móviles	2005	Sí
tel	Detalles de contacto	2005	Sí
travel	Industria de viajes	2005	Sí
xxx	Industria del sexo	2010	No

Figura 7-2. Dominios genéricos de nivel superior.

Los nombres de dominio pueden ser absolutos o relativos. Un nombre de dominio absoluto siempre termina con un punto (por ejemplo, *eng.cisco.com.*), mientras que uno relativo no. Los nombres relativos se tienen que interpretar en cierto contexto para determinar de manera única su significado verdadero. En ambos casos, un dominio nombrado hace referencia a un nodo específico del árbol y a todos los nodos por debajo de él.

Los nombres de dominio no hacen distinción entre mayúsculas y minúsculas, por lo que *edu*, *Edu* y *EDU* significan lo mismo. Los nombres de componentes pueden ser de hasta 63 caracteres de longitud, y los de ruta completa no deben exceder de 255 caracteres.

En principio, los dominios se pueden insertar en el árbol en los dominios genéricos o de países. Por ejemplo, *cs.washington.edu* también podría estar listado bajo el dominio de país *us* como *cs.washington.wa.us*. Sin embargo, en la práctica, casi todas las organizaciones de Estados Unidos están bajo dominios genéricos, y la mayoría de las que están fuera de dicha nación se encuentran bajo el dominio de su país. No hay ninguna regla que impida registrarse bajo dos dominios de nivel superior. Es común que las grandes empresas lo hagan (por ejemplo, *sony.com*, *sony.net* y *sony.nl*).

Cada dominio controla cómo se asignan los dominios que están debajo de él. Por ejemplo, Japón tiene los dominios *ac.jp* y *co.jp* que son espejos de *edu* y *com*. Los Países Bajos no hacen esta distinción y

ponen a todas las organizaciones directamente bajo *nl*. Por lo tanto, los siguientes tres son Departamentos universitarios de Ciencias Computacionales:

1. *cs.washington.edu* (Universidad de Washington, en Estados Unidos).
2. *cs.vu.nl* (Vrije Universiteit, en los Países Bajos).
3. *cs.keio.ac.jp* (Universidad Keio, en Japón).

Para crear un nuevo dominio, se requiere el permiso del dominio en el que se incluirá. Por ejemplo, si se inicia un grupo VLSI en la Universidad de Washington y quieren que se le conozca como *vlsi.cs.washington.edu*, hay que pedir permiso al administrador de *cs.washington.edu*. De la misma forma, si se crea una nueva universidad, digamos la Universidad del Norte de Dakota del Sur, hay que solicitar al administrador del dominio *edu* que le asigne *unds.edu* (si aún está disponible). De esta manera se evitan los conflictos de nombres y cada dominio puede llevar el registro de todos sus subdominios. Una vez que se crea y registra un nuevo dominio, se pueden crear subdominios bajo este nuevo dominio, como *cs.unsd.edu*, sin tener que pedir permiso a nadie que se encuentre a un nivel más alto en el árbol.

Los nombres reflejan los límites organizacionales, no las redes físicas. Por ejemplo, si los Departamentos de Ciencias Computacionales e Ingeniería Eléctrica se ubican en el mismo edificio y comparten la misma LAN, de todas maneras pueden tener dominios diferentes. De manera similar, si el Departamento de Ciencias Computacionales está dividido entre el edificio Babbage y el edificio Turing, por lo general todos los hosts de ambos edificios pertenecerán al mismo dominio.

7.1.2 Registros de recursos de dominio

Cada dominio, sea un host individual o un dominio de nivel superior, puede tener un grupo de **registros de recursos** asociados a él. Estos registros son la base de datos del DNS. En un host individual, el registro de recursos más común es simplemente su dirección IP, pero también existen muchos otros tipos de registros de recursos. Cuando un resolvedor asigna un nombre de dominio al DNS, lo que recibe son los registros de recursos asociados a ese nombre. Por lo tanto, la función principal del DNS es relacionar los nombres de dominios con los registros de recursos.

Un registro de recursos es una 5-tupla. Aunque éstas se codifican en binario por cuestión de eficiencia, en la mayoría de las presentaciones, los registros de recursos se presentan como texto ASCII, una línea por registro de recursos. El formato que usaremos es el siguiente:

```
Nombre_dominio    Tiempo_de_vida    Clase    Tipo    Valor
```

El *Nombre_dominio* indica el dominio al que pertenece este registro. Por lo general, existen muchos registros por dominio y cada copia de la base de datos contiene información sobre múltiples dominios. Por lo tanto, este campo es la clave primaria de búsqueda que se utiliza para atender las consultas. El orden de los registros de la base de datos no es importante.

El campo de *Tiempo_de_vida* es una indicación de la estabilidad del registro. La información altamente estable recibe un valor grande, como 86 400 (la cantidad de segundos en 1 día). A la información que es altamente volátil se le asigna un valor pequeño, como 60 (1 minuto). Regresaremos a este punto después de estudiar el uso de caché.

El tercer campo de cada registro de recursos es la *Clase*. Para información de Internet, siempre es *IN*. Para información que no es de Internet se pueden utilizar otros códigos, pero en la práctica, éstos raras veces se ven.

El campo *Tipo* indica el tipo de registro del que se trata. Hay muchos tipos de registros de DNS. En la figura 7-3 se listan los más importantes.

Tipo	Significado	Valor
SOA	Inicio de autoridad	Parámetros para esta zona.
A	Dirección IPv4 de un host	Entero de 32 bits.
AAAA	Dirección IPv6 de un host	Entero de 128 bits.
MX	Intercambio de correo	Prioridad, dominio dispuesto a aceptar correo electrónico.
NS	Servidor de nombres	Nombre de un servidor para este dominio.
CNAME	Nombre canónico	Nombre de dominio.
PTR	Apuntador	Alias de una dirección IP.
SPF	Marco de trabajo de política del emisor	Codificación de texto de la política de envío de correo.
SRV	Servicio	Host que lo provee.
TXT	Texto	Texto ASCII descriptivo.

Figura 7-3. Los principales tipos de registros de recursos de DNS.

Un registro *SOA* proporciona el nombre de la fuente primaria de información sobre la zona del servidor de nombres (que describiremos más adelante), la dirección de correo electrónico de su administrador, un número de serie único, varias banderas y temporizadores.

El tipo de registro más importante es el registro *A* (dirección), el cual contiene una dirección IPv4 de 32 bits de una interfaz para algún host. El registro *AAAA* correspondiente, o "quad-A", contiene una dirección IPv6 de 128 bits. Cada host de Internet debe tener cuando menos una dirección IP, para que otras máquinas se puedan comunicar con él. Algunos hosts tienen dos o más interfaces de red, en cuyo caso tendrán dos o más registros de recursos tipo *A* o *AAAA*. En consecuencia, DNS puede regresar varias direcciones para un solo nombre.

El registro *MX* es un tipo de registro común que especifica el nombre del host que está preparado para aceptar correo electrónico del dominio especificado. Se utiliza porque no todas las máquinas están preparadas para aceptar correo electrónico. Si alguien desea enviar correo electrónico a, por ejemplo, *bill@microsoft.com*, el host emisor necesita encontrar un servidor de correo en *microsoft.com* que esté dispuesto a aceptar correo electrónico. El registro MX puede proporcionar esta información.

NS es otro tipo de registro importante, ya que especifica un servidor de nombres para el dominio o subdominio. Es un host que tiene una copia de la base de datos para un dominio. Se utiliza como parte del proceso para buscar nombres, el cual describiremos en breve.

Los registros *CNAME* permiten la creación de alias. Por ejemplo, una persona familiarizada con los nombres de Internet en general, que desee enviar un mensaje al usuario *Paul* en el Departamento de Ciencias Computacionales del MIT, podría suponer que *paul@cs.mit.edu* funcionará. No obstante, esta dirección no funcionará, puesto que el dominio del Departamento de Ciencias Computacionales del MIT es *csail.mit.edu*. Sin embargo, como servicio para la gente que no sabe esto, el MIT podría crear una entrada *CNAME* para encaminar a la gente y a los programas en la dirección correcta. La entrada podría ser como la siguiente:

```
cs.mit.edu      86400    IN    CNAME    csail.mit.edu
```

Al igual que *CNAME*, *PTR* apunta a otro nombre. Sin embargo, a diferencia de *CNAME*, que en realidad es sólo una definición macro (es decir, un mecanismo para reemplazar una cadena por otra), *PTR* es un tipo de datos DNS normal, cuya interpretación depende del contexto en el que se encuentre. En la práctica,

casi siempre se usa para asociar un nombre a una dirección IP con el fin de permitir búsquedas de la dirección IP y devolver el nombre de la máquina correspondiente. Éstas se llaman **búsquedas inversas**.

SRV es un tipo reciente de registro, el cual permite identificar a un host para un servicio dado en un dominio. Por ejemplo, el servidor web para *cs.washington.edu* se podría identificar como *cockatoo.cs.washington.edu*. Este registro generaliza el registro *MX* que realiza la misma tarea, pero que es sólo para servidores de correo.

SPF es otro tipo reciente de registro, el cual permite a un dominio codificar información sobre qué máquinas en el dominio enviarán correo al resto de Internet. Esto ayuda a las máquinas receptoras a verificar que el correo sea válido. Si se está recibiendo correo de una máquina que se identifique como *dodgy*, pero el registro indica que el correo sólo se enviará fuera del dominio por medio de una máquina llamada *smtp*, es probable que sea correo basura falsificado.

Al final de la lista se encuentran los registros *TXT*, cuyo objetivo original era permitir que los dominios se identificaran a sí mismos en formas arbitrarias. En la actualidad usualmente codifican información legible por máquina, por lo general la información *SPF*.

Por último, tenemos el campo *Valor*. Este campo puede ser un número, un nombre de dominio o una cadena ASCII. La semántica depende del tipo de registro. En la figura 7-3 se proporciona una descripción corta de los campos *Valor* para cada uno de los tipos principales de registro.

Como ejemplo del tipo de información que podríamos encontrar en la base de datos DNS de un dominio, vea la figura 7-4. En ésta se ilustra una parte de una base de datos (hipotética) para el dominio *cs.vu.nl* que se muestra en la figura 7-1. La base de datos contiene siete tipos de registros de recursos.

La primera línea sin comentario de la figura 7-4 da un poco de información básica sobre el dominio, por lo que no entraremos en detalles. Luego están dos entradas que dan el primero y segundo lugares a donde se pretenderá entregar correo electrónico dirigido a *persona@cs.vu.nl*. Primero se intentará en *zephyr* (una máquina específica). Si falla, a continuación se hará un intento en *top* que es la siguiente opción. La siguiente línea identifica el servidor de correo para el dominio como *star*.

```
; Datos autoritarios para cs.vu.nl
cs.vu.nl.      86400   IN   SOA     star boss (9527, 7200, 7200, 241920, 86400)
cs.vu.nl.      86400   IN   MX      1 zephyr
cs.vu.nl.      86400   IN   MX      2 top
cs.vu.nl.      86400   IN   NS      star

star           86400   IN   A       130.37.56.205
zephyr         86400   IN   A       130.37.20.10
top            86400   IN   A       130.37.20.11
www            86400   IN   CNAME   star.cs.vu.nl
ftp            86400   IN   CNAME   zephyr.cs.vu.nl

flits          86400   IN   A       130.37.16.112
flits          86400   IN   A       192.31.231.165
flits          86400   IN   MX      1 flits
flits          86400   IN   MX      2 zephyr
flits          86400   IN   MX      3 top

rowboat                IN   A       130.37.56.201
                       IN   MX      1 rowboat
                       IN   MX      2 zephyr

little-sister          IN   A       130.37.62.23

laserjet               IN   A       192.31.231.216
```

Figura 7-4. Parte de una posible base de datos DNS para *cs.vu.nl*.

Después de la línea en blanco (que se agregó para mejorar la legibilidad), siguen líneas que proporcionan las direcciones IP para *star*, *zephyr* y *top*. Estas direcciones van seguidas de un alias, *www.vs.vu.nl*, para usar esta dirección sin designar una máquina específica. La creación de este alias permite a *cs.vu.nl* cambiar su servidor web sin invalidar la dirección que la gente usa para llegar a él. Se aplica un argumento similar para *ftp.cs.vu.nl*.

La sección para la máquina *flits* lista dos direcciones IP y se proporcionan tres opciones para manejar el correo electrónico que se envía a *flits.cs.vu.nl*. Lógicamente la primera opción es la misma *flits*, pero si está desconectada, *zephyr* y *top* son la segunda y tercera opciones, respectivamente.

Las siguientes tres líneas contienen una entrada común para una computadora; en este caso, *rowboat.cs.vu.nl*. La información que se proporciona contiene la dirección IP junto con los destinos de correo primario y secundario. Después viene una entrada para una computadora que no es capaz de recibir correo por sí sola, seguida de una entrada que quizá sea para una impresora conectada a Internet.

7.1.3 Servidores de nombres

Cuando menos en teoría, un solo servidor de nombres podría contener toda la base de datos DNS y responder a todas las consultas relacionadas con ella. En la práctica, este servidor estaría tan sobrecargado que sería inservible. Además, si llegara a fallar, Internet completa quedaría inutilizable.

Para evitar los problemas asociados al hecho de tener una sola fuente de información, el espacio de nombres DNS se divide en **zonas** que no se sobreponen. La figura 7-5 muestra una manera posible de dividir el espacio de nombres de la figura 7-1. Cada zona circulada contiene una parte del árbol.

El lugar donde se colocan los límites dentro de una zona es responsabilidad del administrador de esa zona. Esta decisión se toma en gran parte con base en la cantidad de servidores de nombres deseados y su ubicación. Por ejemplo, en la figura 7-5 la Universidad de Washington tiene una zona para *washington.edu* que maneja *eng.washington.edu*, pero no maneja *cs.washington.edu*. Ésta es una zona separada con sus propios servidores de nombres. Tal decisión podría tomarse cuando un Departamento, como el de Inglés, no desee operar su propio servidor de nombres, pero un Departamento como el de Ciencias Computacionales sí.

Cada zona se asocia también con uno o más servidores de nombres. Éstos son hosts que contienen la base de datos de la zona. Por lo general, una zona debe tener un servidor de nombres primario, el cual obtiene su información de un archivo en su disco, y uno o más servidores secundarios, los cuales obtienen

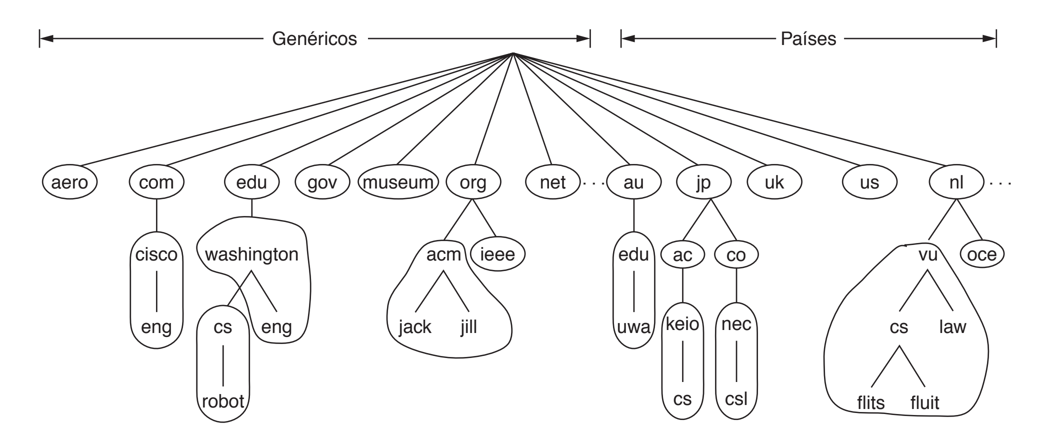


Figura 7-5. Parte del espacio de nombres DNS dividido en zonas (la cuales están circuladas).

su información del servidor de nombres primario. Para mejorar la confiabilidad, algunos de los servidores de nombres se pueden ubicar fuera de la zona.

Al proceso de buscar un nombre y encontrar una dirección se le conoce como **resolución de nombres**. Cuando un resolvedor tiene una consulta referente a un nombre de dominio, la pasa a uno de los servidores de nombres locales. Si el dominio que se busca está bajo la jurisdicción del servidor de nombres, como *top.cs.vu-nl* que está bajo *cs.vu.nl*, devuelve los registros de recursos autorizados. Un **registro autorizado** es uno que proviene de la autoridad que administra a ese registro y, por ende, siempre es correcto. Los registros autorizados contrastan con los **registros en caché**, que tal vez no estén actualizados.

¿Qué pasa cuando el dominio es remoto, como cuando *flits.cs.vu.nl* desea encontrar la dirección IP de *robot.cs.washington.edu* en la UW (Universidad de Washington)? En este caso y si no hay información en la caché sobre el dominio disponible en forma local, el servidor de nombres empieza una consulta remota. Esta consulta sigue el proceso que se muestra en la figura 7-6. El paso 1 muestra la consulta que se envía al servidor de nombres local. Esta consulta contiene el nombre de dominio buscado, el tipo (*A*) y la clase (*IN*).

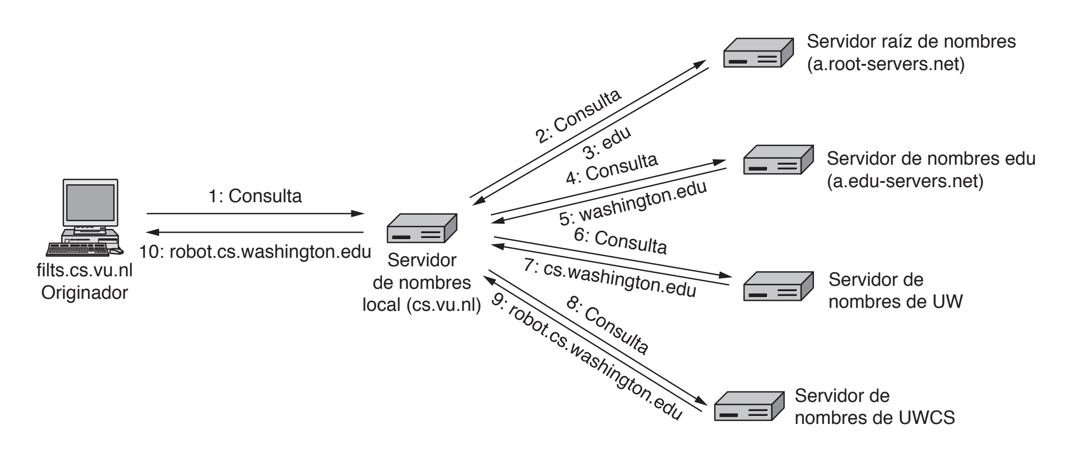


Figura 7-6. Ejemplo de un resolvedor que busca un nombre remoto en 10 pasos.

El siguiente paso es empezar en la parte superior de la jerarquía de nombres y preguntar a cada uno de los **servidores raíz de nombres**. Estos servidores tienen información sobre cada dominio de nivel superior. Esto se muestra como el paso 2 en la figura 7-6. Para contactarse con un servidor raíz, cada servidor de nombres debe tener información sobre uno o más servidores raíz de nombres. Esta información por lo general está presente en un archivo de configuración del sistema que se carga en la caché DNS al momento de iniciar el servidor DNS. En esencia, es una lista de registros *NS* para la raíz y los registros *A* correspondientes.

Hay 13 servidores raíz DNS, que por falta de imaginación se llaman *a-root-servers.net* hasta *m.root-servers.net*. Cada servidor raíz podría ser lógicamente una sola computadora. Sin embargo, como toda Internet depende de los servidores raíz, éstos son computadoras poderosas con un alto grado de replicación. La mayoría de los servidores están presentes en varias ubicaciones geográficas y se puede acceder a ellos mediante el enrutamiento *anycast*, en donde un paquete se entrega a la instancia más cercana de una dirección de destino; en el capítulo 5 vimos el enrutamiento *anycast*. La replicación mejora la confiabilidad y el desempeño.

Es improbable que el servidor raíz de nombres conozca la dirección de una máquina en la UW, y tal vez tampoco conozca el servidor de nombres para la UW. Pero debe conocer el servidor de nombres del dominio *edu*, en donde se encuentra *cs.washington.edu*. En el paso 3 devuelve el nombre y la dirección IP para esa parte de la respuesta.

Después, el servidor de nombres local continúa su búsqueda y envía la consulta completa al servidor de nombres *edu* (*a.edu-servers.net*). Ese servidor de nombres devuelve el servidor de nombres para la UW. Esto se muestra en los pasos 4 y 5. Ahora que está más cerca, el servidor de nombres local envía la consulta al servidor de nombres de la UW (paso 6). Si el nombre de dominio buscado estaba en el Departamento de Inglés se hubiera encontrado la respuesta, puesto que la zona de la UW incluye a dicho Departamento. Pero el Departamento de Ciencias Computacionales decidió implementar su propio servidor de nombres. La consulta devuelve el nombre y la dirección IP del servidor de nombres de Ciencias Computacionales de la UW (paso 7).

Por último, el servidor de nombres local consulta al servidor de nombres de Ciencias Computacionales de la UW (paso 8). Este servidor es autorizado para el dominio *cs.washington.edu*, por lo que debe tener la respuesta. A continuación devuelve la respuesta final (paso 9), que el servidor de nombres local reenvía como respuesta a *flits.cs.vu.nl* (paso 10). Así termina la resolución del nombre.

Podemos explorar este proceso mediante el uso de herramientas estándar, como el programa *dig* que está instalado en la mayoría de los sistemas UNIX. Por ejemplo, al escribir

```
dig @a.edu-servers-net robot.cs.washington.edu
```

se enviará una consulta sobre *robot.cs.washington.edu* al servidor de nombres *a.edu-servers.net* y se imprimirá el resultado en pantalla. Esto nos mostrará la información obtenida en el paso 4 del ejemplo anterior, y conoceremos el nombre y la dirección IP de los servidores de nombres de la UW.

Hay tres puntos técnicos a discutir sobre este gran escenario. Primero, en la figura 7-6 se utilizan dos mecanismos de consulta distintos. Cuando el host *flits.cs.vu.nl* envía su consulta al servidor de nombres local, ese servidor maneja la resolución en representación de *flits* hasta que pueda devolver la respuesta deseada. *No* devuelve respuestas parciales. Podrían ser útiles, pero no es lo que se buscaba. A este mecanismo se le denomina **consulta recursiva**.

Por otra parte, el servidor raíz de nombres (y cada servidor de nombres subsiguiente) no continúa de manera recursiva la consulta para el servidor de nombres local. Sólo devuelve una respuesta parcial y avanza a la siguiente consulta. El servidor de nombres local es responsable de continuar con la resolución, para lo cual emite otras consultas adicionales. A este mecanismo se le denomina **consulta iterativa**.

La resolución de un nombre puede implicar ambos mecanismos, como se mostró en el ejemplo anterior. Tal vez siempre parezca que es preferible una consulta recursiva, pero muchos servidores (en especial el servidor raíz) no las manejarán debido a que están muy ocupados. Las consultas iterativas imponen la carga en quien las originó. La razón de que el servidor de nombres local soporte una consulta recursiva es debido a que proporciona un servicio a los hosts en su dominio. Estos hosts no tienen que configurarse para ejecutar un servidor de nombres completo, sólo para acceder al servidor de nombres local.

El segundo punto es la caché. Todas las respuestas, incluyendo las que se devuelven, se colocan en la caché. De esta manera, si otro host *cs.vu.nl* genera una consulta sobre *robot.cs.washington.edu*, ya se sabrá la respuesta. Incluso, si un host genera una consulta sobre un host distinto en el mismo dominio, por decir *galah.cs.washington.edu*, esta consulta se puede enviar directamente al servidor de nombres autorizado. Asimismo, las consultas sobre otros dominios en *washington.edu* pueden empezar desde el servidor de nombres de *washington.edu*. El uso de respuestas de la caché reduce de manera considerable los pasos en una consulta y mejora el desempeño. El escenario original que bosquejamos es, de hecho, el peor caso que ocurre cuando no se coloca información útil en la caché.

Sin embargo, las respuestas de la caché no son autorizadas debido a que los cambios realizados en *cs.washington.edu* no se propagarán a todas las cachés en el mundo que puedan conocer sobre este nombre. Por esta razón, las entradas de la caché no deben durar mucho tiempo. Esto explica por qué se incluye el campo *Tiempo_de_vida* en cada registro de recursos. Si cierta máquina ha tenido la misma

dirección IP durante años, puede ser seguro colocar esa información en la caché por un día. En el caso de información más volátil, podría ser más seguro purgar los registros después de unos cuantos segundos o de un minuto.

El tercer punto es el protocolo de transporte que se utiliza para las consultas y respuestas: UDP. Los mensajes DNS se envían en paquetes UDP con un formato simple para las consultas, las respuestas y los servidores de nombres que se pueden usar para continuar la resolución. No entraremos en los detalles de este formato. Si no llega una respuesta dentro de un intervalo corto, el cliente DNS repite la consulta e intenta con otro servidor para ese dominio después de unos cuantos reintentos. Este proceso está diseñado para manejar el caso en que el servidor esté fallando, así como cuando se pierde el paquete de consulta o de respuesta. Se incluye un identificador de 16 bits en cada consulta y se copia a la respuesta, de modo que un servidor de nombres pueda relacionar las respuestas con la correspondiente consulta, incluso aunque haya varias respuestas pendientes al mismo tiempo.

Aun cuando su propósito es simple, debe quedar claro que DNS es un sistema distribuido grande y complejo, compuesto por millones de servidores de nombres que trabajan en conjunto. Forma un vínculo clave entre los nombres de dominios legibles por humanos y las direcciones IP de las máquinas. Incluye la replicación y el uso de caché para fines de desempeño y confiabilidad; además está diseñado para ser muy poderoso.

No hemos hablado sobre la seguridad pero, como podría imaginar, la habilidad de cambiar la asignación nombre-dirección puede tener consecuencias devastadoras si se hace de forma malintencionada. Por esta razón se han desarrollado extensiones de seguridad para DNS, conocidas como DNSSEC; las describiremos en el capítulo 8.

También existe una demanda según la aplicación requerida de usar los nombres en formas más flexibles; por ejemplo, al nombrar contenido y resolver, con base en ese contenido, la dirección IP de un host cercano que lo contiene. Eso encaja con el modelo de buscar una película y descargarla. Lo que importa es la película y no la computadora que tiene una copia de ella, por lo que todo lo que queremos es la dirección IP de cualquier computadora cercana que tenga una copia de esa película. Las redes de distribución de contenido son una forma de lograr esta asignación. En la sección 7.5 describiremos la forma en que estas redes se basan en el DNS.

7.2 CORREO ELECTRÓNICO

El **correo electrónico** o **email**, como se le conoce comúnmente, ha existido por más de tres décadas. Puesto que es mucho más rápido y económico que el correo convencional, el correo electrónico ha sido una aplicación popular desde los primeros días de Internet. Antes de 1990 se utilizaba principalmente en ambientes académicos. En la década de 1990 se dio a conocer al público en general y creció en forma exponencial, al punto que el número de mensajes de correo electrónico enviados por día ahora es mucho mayor que el número de cartas por correo fuera de línea (es decir, en papel). Existen otras formas de comunicación en red, como la mensajería instantánea y las llamadas de voz sobre IP, cuyo uso se ha expandido en forma considerable durante la última década, pero el correo electrónico sigue siendo el caballo de batalla de la comunicación en Internet. Se utiliza mucho dentro de la industria para la comunicación entre empresas; por ejemplo, para permitir que los empleados distantes de todo el mundo cooperen en proyectos complejos. Por desgracia y al igual que el correo convencional, la mayoría del correo electrónico (aproximadamente 9 de cada 10 mensajes) es correo basura o **spam** (McAfee, 2010).

El correo electrónico, como la mayoría de otras formas de comunicación, ha desarrollado sus propias convenciones y estilos. Es muy informal y tiene un umbral bajo de uso. Las personas que nunca hubieran soñado con escribir una carta a un personaje importante, no dudarían un instante para enviarle un

mensaje de correo electrónico. Al eliminar la mayoría de las indicaciones asociadas con el rango, la edad y el género, los debates por correo electrónico se enfocan por lo regular en el contenido y no en el estatus. Así, una idea brillante de un estudiante de verano puede tener más impacto que una idea tonta de un vicepresidente ejecutivo.

El correo electrónico está lleno de abreviaturas, como SYL (*See You Later*, en español: nos vemos luego), FYI (*For Your Information*, en español: para su información) o TQM (Te Quiero Mucho). Muchas personas también utilizan pequeños símbolos ASCII llamados **caritas** (emoticones), que empiezan con el ubicuo ":-)". (Si no le es familiar este símbolo, gire el libro 90 grados en sentido de las manecillas del reloj). Este símbolo y los demás **emoticonos** ayudan a transmitir el tono del mensaje. Estos símbolos se han esparcido a otras formas lacónicas de comunicación, como la mensajería instantánea.

Los protocolos de correo electrónico también han evolucionado durante el periodo en que se han utilizado. Los primeros sistemas simplemente consistían en protocolos de transferencia de archivos, con la convención de que la primera línea de cada mensaje (es decir, archivo) contenía la dirección del destinatario. A medida que pasó el tiempo, el correo electrónico se separó de la transferencia de archivos y se agregaron muchas características, como la habilidad de enviar un mensaje a una lista de contactos. Las herramientas multimedia se hicieron populares en 1990 para enviar mensajes con imágenes y otros materiales además del texto. Los programas para leer correo electrónico se hicieron mucho más sofisticados también, pues cambiaron de una interfaz basada en texto a las interfaces gráficas de usuario, además de agregar la capacidad de que los usuarios accedieran a su correo desde sus laptops en donde quiera que se encontraran. Finalmente, con la prevalencia del correo spam, ahora los lectores de correo y los protocolos de transferencia de correo deben poner atención, a cómo detectar y eliminar el correo electrónico no deseado.

En nuestra descripción del correo electrónico, nos enfocaremos en la forma en que se desplazan los mensajes de un usuario a otro, en vez de la apariencia visual de los programas para leer un correo. Sin embargo, después de escribir la arquitectura en general, empezaremos con la parte correspondiente a la interfaz de usuario del sistema de correo electrónico, pues la mayoría de los lectores están familiarizados en esta cuestión.

7.2.1 Arquitectura y servicios

En esta sección ofreceremos un panorama de la manera como están organizados los sistemas de correo electrónico y lo que pueden hacer. En la figura 7-7 se muestra la arquitectura del sistema, que consiste en dos tipos de subsistemas: los **agentes de usuario**, que permiten a la gente leer y enviar correo electrónico, y los **agentes de transferencia de mensajes**, que mueven los mensajes del origen al destino. También nos referiremos a los agentes de transferencia de mensajes de una manera informal como **servidores de correo**.

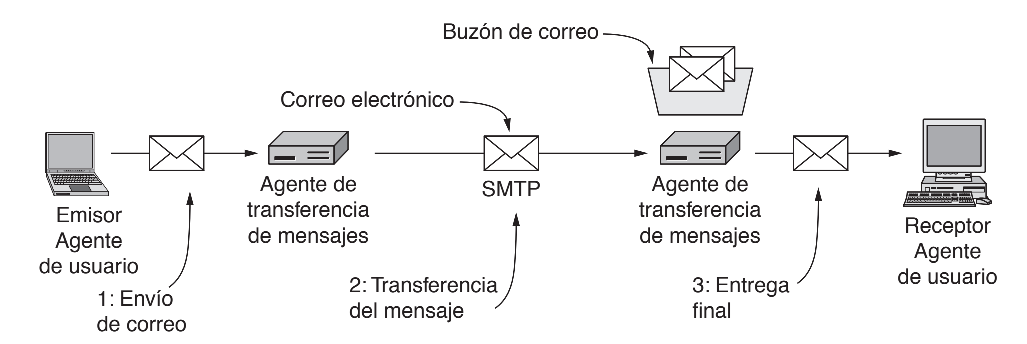


Figura 7-7. Arquitectura del sistema de correo electrónico.

El agente de usuario es un programa que proporciona una interfaz gráfica, o algunas veces una interfaz basada en texto y comandos, que permite interactuar con el sistema de correo electrónico. Incluye un medio para redactar mensajes y respuestas a éstos, desplegar los mensajes entrantes y organizarlos al archivarlos, realizar búsquedas y eliminarlos. El acto de enviar nuevos mensajes al sistema de correo para su entrega se conoce como **envío de correo**.

Una parte del procesamiento de los agentes de usuario se puede realizar de manera automática para anticiparse a lo que desea el usuario. Por ejemplo, el correo entrante se puede filtrar para extraer o quitar prioridad a los mensajes que quizá sean spam. Algunos agentes de usuario incluyen características avanzadas, como las respuestas automáticas por correo electrónico ("Estoy disfrutando unas vacaciones maravillosas y me tardaré un poco en contestarle"). Un agente de usuario se ejecuta en la misma computadora en la que un usuario lee su correo. Es sólo otro programa más y se puede ejecutar sólo parte del tiempo.

Los agentes de transferencia de mensajes son en general procesos del sistema. Se ejecutan en segundo plano en equipos servidores de correo con la intención de estar siempre disponibles. Su tarea es mover de manera automática el correo electrónico a través del sistema, desde el que lo originó hasta el receptor mediante el **SMTP** (**Protocolo Simple de Transferencia de Correo**, del inglés *Simple Mail Transfer Protocol*). Éste es el paso de transferencia del mensaje.

El SMTP se especificó originalmente como el RFC 821 y se revisó para convertirse en el RFC 5321 actual. Envía el correo electrónico sobre las conexiones y devuelve el estado de entrega, junto con los errores. Existen muchas aplicaciones en donde es importante la confirmación de la entrega, e incluso puede tener un significado legal ("Bueno, su señoría, mi sistema de correo electrónico no es muy confiable, así que creo que el citatorio electrónico simplemente se perdió por ahí").

Los agentes de transferencia de mensajes también implementan las **listas de correo**, en donde se entrega una copia idéntica de un mensaje a cualquiera de los miembros en una lista de direcciones de correo electrónico. Otras características avanzadas son las copias al carbón, copias al carbón ocultas, correo electrónico de alta prioridad, correo electrónico secreto (es decir, cifrado), destinatarios alternativos si el primario no está disponible en ese momento, y la habilidad de que los asistentes lean y respondan el correo electrónico de sus jefes.

Vincular los agentes de usuario y los agentes de transferencia de mensajes son los conceptos de los buzones de correo y un formato estándar de mensajes de correo electrónico. Los **buzones de correo** almacenan el correo electrónico que recibe un usuario. Los servidores de correo se encargan de su mantenimiento. Para ello, los agentes de usuario envían comandos a los servidores de correo para manipular los buzones de correo, inspeccionar su contenido, eliminar mensajes, etc. La recuperación del correo es la entrega final (paso 3) en la figura 7-7. Con esta arquitectura, un usuario puede usar distintos agentes de usuario en varias computadoras para acceder a un buzón de correo.

El correo se envía entre un agente de transferencia de mensajes y otro en un formato estándar. El formato original, RFC 822, se revisó para convertirse en el RFC 5322 actual, que se extendió con soporte para contenido multimedia y texto internacional. A este esquema se le conoce como MIME y hablaremos de él más adelante. A pesar de esta actualización, la gente aún se refiere al correo electrónico de Internet como RFC 822.

Una idea clave en el formato de los mensajes es la distinción entre la **envoltura** y su contenido. La primera encapsula el mensaje y contiene toda la información necesaria para transportarlo, como la dirección de destino, la prioridad y el nivel de seguridad, todo lo cual es distinto del mensaje en sí. Los agentes de transporte de mensajes usan la envoltura para el enrutamiento, al igual que la oficina postal con el correo convencional.

El mensaje dentro de la envoltura consiste en dos partes separadas: el **encabezado** y el **cuerpo**. El primero contiene la información de control para los agentes de usuario. El cuerpo es exclusivo para el destinatario humano; a ninguno de los agentes le importa mucho. En la figura 7-8 se ilustran las envolturas y los mensajes.

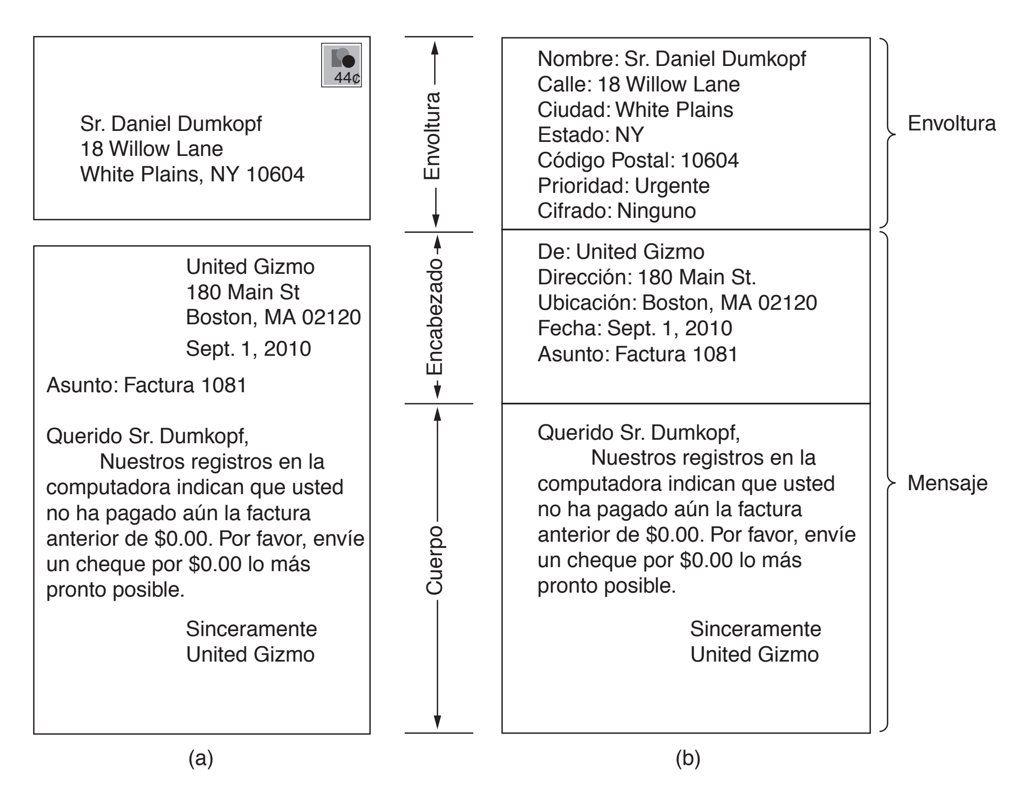


Figura 7-8. Envolturas y mensajes. (a) Correo convencional. (b) Correo electrónico.

Examinaremos las piezas de esta arquitectura con más detalle, para lo cual analizaremos los pasos involucrados en el envío de correo electrónico de un usuario a otro. Este trayecto empieza con el agente de usuario.

7.2.2 El agente de usuario

Un agente de usuario es un programa (algunas veces conocido como **lector de correo electrónico**) que acepta una variedad de comandos para redactar mensajes, recibirlos y responder a ellos, así como para manipular buzones de correo. Existen muchos agentes de usuario populares, incluyendo Google Gmail, Microsoft Outlook, Mozilla Thunderbird y Apple Mail. Su apariencia puede variar de manera considerable. La mayoría de los agentes de usuario tienen una interfaz gráfica controlada por menús o iconos, en la cual se requiere un ratón o una interfaz táctil en los dispositivos móviles más pequeños. Los agentes de usuario antiguos, como Elm, mh y Pine, proveen interfaces basadas en texto y esperan que el usuario introduzca comandos de un solo carácter desde el teclado. En sentido funcional todos son iguales, al menos para los mensajes de texto.

En la figura 7-9 se muestran los elementos típicos de una interfaz de agente de usuario. Es probable que su lector de correo sea más llamativo, pero es casi seguro que tenga funciones equivalentes. Al iniciar un agente de usuario, por lo general presenta un resumen de los mensajes en el buzón de correo del usuario. Es común que el resumen muestre una línea para cada mensaje en cierto orden y que resalte los campos clave del mensaje que se extraen de su envoltura o encabezado.

En el ejemplo de la figura 7-9 se muestran siete líneas de resumen. Éstas usan los campos *De*, *Asunto* y *Recibido* en ese orden para mostrar quién envió el mensaje, de qué trata y cuándo se recibió. Se aplica

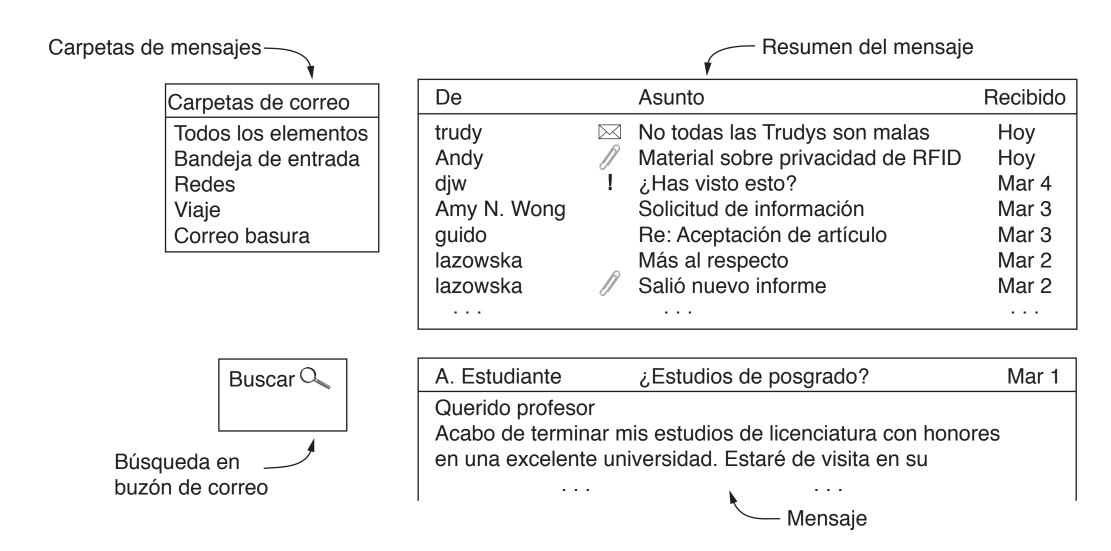


Figura 7-9. Elementos comunes de la interfaz de agente de usuario.

formato a toda la información de una manera amigable para el usuario, en vez de mostrar el contenido literal de los campos de mensaje, pero se basa en estos campos. Así, las personas que no incluyen un campo *Asunto* descubren a menudo que las respuestas a sus correos electrónicos tienden a no recibir la prioridad más alta.

Hay muchos otros campos o indicaciones posibles. Por ejemplo, los iconos enseguida de los asuntos de los mensajes en la figura 7-9 podrían indicar correo sin leer (la envoltura), material adjunto (el clip de papel) y correo importante, según lo juzgue el emisor (el signo de exclamación).

También hay muchas formas posibles de ordenar los mensajes. La más común es ordenarlos con base en la hora y fecha en que se recibieron: primero los más recientes, con alguna indicación acerca de si el mensaje es nuevo o el usuario ya lo leyó. El usuario puede personalizar los campos en el resumen y la forma de ordenarlos, de acuerdo con sus preferencias.

Los agentes de usuario también deben ser capaces de mostrar los mensajes entrantes según sea necesario, de modo que las personas puedan leer su correo electrónico. A menudo se provee una vista previa del mensaje, como en la figura 7-9, para ayudar a los usuarios a decidir cuándo es necesario entrar en más detalles. Estas vistas pueden usar pequeños iconos o imágenes para describir el contenido del mensaje. Otro tipo de procesamiento de la presentación incluye cambiar el formato a los mensajes para que se ajusten a la pantalla, además de traducir o convertir el contenido a formatos más convenientes (por ejemplo, voz digitalizada a texto reconocido).

Una vez que el usuario lee un mensaje, puede decidir qué hacer con él. A esto se le conoce como **disposición del mensaje**. Las opciones incluyen eliminarlo, enviar una respuesta, reenviarlo a otro usuario y guardarlo para después hacer referencia a él. La mayoría de los agentes de usuario pueden administrar un buzón de correo para el entrante con varias carpetas para el que se encuentra almacenado. Estas carpetas permiten al usuario guardar los mensajes de acuerdo con el emisor, el asunto o cualquier otra categoría.

El proceso de archivar los mensajes también lo puede realizar el agente de usuario en forma automática, antes de que el usuario lea los mensajes. Un ejemplo común es que los campos y el contenido de los mensajes se inspeccionan y utilizan, junto con la retroalimentación del usuario sobre los mensajes anteriores, para determinar si es probable que un mensaje sea spam. Muchos ISP y empresas ejecutan software que etiqueta el correo como importante o como basura, para que el agente de usuario lo pueda archivar en

el correspondiente buzón de correo. El ISP y la empresa tienen la ventaja de que manejan el correo para muchos usuarios y pueden tener listas de *spammers* conocidos. Si cientos de usuarios reciben un mensaje similar, probablemente sea spam. Al ordenar con anterioridad el correo entrante como "posiblemente legítimo" y "posiblemente spam", el agente de usuario puede ahorrar a los usuarios una cantidad considerable del trabajo relacionado con separar el correo bueno del considerado como basura.

¿Y el spam más popular? Lo generan grupos de computadoras comprometidas llamadas **botnets**; su contenido depende de la ubicación del usuario que lee el correo. Los diplomas falsos son comunes en Asia; los fármacos baratos y otras ofertas de dudosa procedencia son comunes en Estados Unidos. Las cuentas de banco nigerianas no reclamadas siguen abundando. Las píldoras para engrandecer ciertas partes del cuerpo son comunes en todas partes.

Los usuarios pueden construir otras reglas para archivar mensajes. Cada regla especifica una condición y una acción. Por ejemplo, una regla podría establecer que cualquier mensaje recibido del jefe debe ir a una carpeta para leerlo de inmediato, y cualquier mensaje de una lista de correo específica debe pasar a otra carpeta para que se lea después. En la figura 7-9 se muestran varias carpetas. Las más importantes son la Bandeja de entrada (*Inbox*), para el correo entrante que no se archiva en ninguna carpeta específica, y el Correo basura (*Junk Mail*) para los mensajes que pueden serlo.

Además de construcciones explícitas como las carpetas, ahora los agentes de usuario proporcionan herramientas completas para buscar información en el buzón de correo. Esta característica también se muestra en la figura 7-9. Las herramientas de búsqueda permiten a los usuarios encontrar mensajes con rapidez, como el mensaje acerca de "dónde comprar Vegemite" que alguien envió el mes pasado.

El correo electrónico ha mostrado grandes avances desde los días en que sólo eran transferencias de archivos. Ahora los proveedores soportan de manera rutinaria buzones de hasta 1 GB de correo almacenado, que contiene los detalles de las interacciones de un usuario durante un largo periodo. La sofisticación con que los agentes de usuario manejan el correo, con herramientas de búsqueda y formas automáticas de procesamiento, es lo que hace posible administrar estos grandes volúmenes de correo electrónico. Para las personas que envían y reciben miles de mensajes al año, estas herramientas son invaluables.

Otra herramienta útil es la habilidad de responder de manera automática a los mensajes en cierta forma. Una respuesta es reenviar el correo electrónico entrante a una dirección distinta; por ejemplo, una computadora operada por un servicio comercial de radiolocalización que ubica al usuario mediante el uso de radio o satélite, y muestra la línea *Asunto:* en su radiolocalizador. Estos **autorrespondedores** se deben ejecutar en el servidor de correo, ya que el agente de usuario tal vez no se ejecute todo el tiempo y sólo puede recuperar el correo electrónico en ciertas ocasiones. Debido a estos factores, tal vez el agente de usuario no pueda proveer una verdadera respuesta automática. Sin embargo, por lo general el agente de usuario presenta la interfaz para las respuestas automáticas.

El **agente de vacaciones** es un ejemplo distinto de respuesta automática. Éste es un programa que examina cada mensaje entrante y envía al emisor una respuesta grabada como: "Hola. Estoy de vacaciones. Regresaré el 24 de agosto. Hablaremos hasta entonces". Dichas respuestas también pueden especificar cómo manejar asuntos urgentes en el intermedio, otras personas a quienes se pueda contactar en caso de problemas específicos, etc. La mayoría de los agentes de vacaciones lleva un registro de a quién le ha enviado respuestas grabadas y se abstiene de enviar a la misma persona una segunda respuesta. Sin embargo, hay algunas desventajas al usar estos agentes. Por ejemplo, no es aconsejable enviar una respuesta grabada a una lista de correo extensa.

Veamos ahora el escenario en que un usuario envía un mensaje a otro. Una de las características básicas que soportan los agentes de usuario y que aún no hemos examinado es la redacción del correo. Esta característica involucra la creación de mensajes y respuestas a éstos, además del envío de esos mensajes al resto del sistema de correo para entregarlos a su destino. Aunque podemos usar cualquier editor de texto para crear el cuerpo del mensaje, por lo general los editores están integrados al agente de usuario, de modo que pueda proveer asistencia con las direcciones y los diversos campos de encabezado

que se incluyen en cada mensaje. Por ejemplo, al responder un mensaje el sistema de correo electrónico puede extraer la dirección del originador del correo electrónico entrante, e insertarla de manera automática en el lugar apropiado en la respuesta. Otras características comunes son adjuntar un **bloque de firma** a la parte inferior de un mensaje, corregir la ortografía y calcular firmas digitales que muestren que el mensaje es válido.

Los mensajes que se envían al sistema de correo tienen un formato estándar que se debe crear a partir de la información que se provee al agente de usuario. La parte más importante del mensaje para la transferencia es la envoltura, y la parte más importante de la envoltura es la dirección de destino. Esta dirección debe estar en un formato con el que puedan lidiar los agentes de transferencia de mensajes.

La forma esperada de una dirección es *usuario@dirección-dns*. Puesto que estudiamos el DNS anteriormente en este capítulo, no repetiremos ese material aquí. Sin embargo, vale la pena mencionar que existen otras formas de direccionamiento. En particular, las direcciones **X.400** tienen una apariencia muy distinta en comparación con las direcciones DNS.

X.400 es un estándar ISO para los sistemas de administración de correo que alguna vez compitió con SMTP. Este último ganó con facilidad, aunque aún se utilizan sistemas X.400, en su mayoría fuera de Estados Unidos. Las direcciones X.400 están compuestas de pares *atributo=valor* separados por barras diagonales; por ejemplo:

/P=US/ES=MASSACHUSETTS/L=CAMBRIDGE/DP=360 MEMORIAL DR./NC=KEN SMITH/

Esta dirección especifica país, estado, localidad, dirección personal y nombre común (Ken Smith). Hay muchos otros atributos posibles, por lo que podemos enviar correo electrónico a alguien cuya dirección exacta no sepamos, siempre y cuando conozcamos suficientes atributos adicionales (por ejemplo, empresa y puesto).

Aunque los nombres X.400 son menos convenientes que los nombres DNS, con los agentes de usuario este aspecto es debatible, puesto que tienen alias amigables para los usuarios (algunas veces conocidos como apodos, o "alias") que les permiten introducir o seleccionar el nombre de una persona y obtener la dirección correcta de correo electrónico. En consecuencia, por lo general no es necesario introducir estas extrañas cadenas de texto.

Un punto final a considerar sobre el envío de correo son las listas de correo, las cuales permiten a los usuarios enviar el mismo mensaje a una lista de personas con un solo comando. Hay dos opciones en cuanto a la forma de dar mantenimiento a la lista de correo. Se podría mantener en forma local, mediante el agente de usuario. En este caso, el agente simplemente envía un mensaje separado a cada contacto deseado.

La otra alternativa es mantener la lista en forma remota con un agente de transferencia de mensajes. Así, los mensajes se expandirán en el sistema de transferencia de mensajes, lo cual tiene el efecto de permitir que varios usuarios envíen información a la lista. Por ejemplo, si un grupo de aficionados a los pájaros tiene una lista de correo llamada *pajareros*, la cual está instalada en el agente de transferencia *meadowlark.arizona.edu*, cualquier mensaje que se envíe a *pajareros@meadowlark.arizona.edu* se enrutará a la Universidad de Arizona y se expandirá en mensajes individuales a todos los miembros de la lista de correos, en cualquier parte del mundo en donde se encuentren. Los usuarios de esta lista no pueden saber que están en una lista de correo. Podría ser tan sólo el buzón de correo personal del profesor Gabriel O. Pajareros.

7.2.3 Formatos de mensaje

Ahora pasaremos de la interfaz de usuario al formato de los mensajes de correo electrónico en sí. Los mensajes que envía el agente de usuario se deben colocar en un formato estándar para ser manejados por

el agente de transferencia de mensajes. Primero veremos las bases del correo electrónico ASCII usando el RFC 5322, que es la revisión más reciente del formato de mensaje de Internet original según lo descrito en el RFC 822. Después de eso analizaremos las extensiones multimedia para el formato básico.

RFC 5322: el formato de mensaje de Internet

Los mensajes consisten en una envoltura primitiva (descrita como parte del SMTP en el RFC 5321), cierto número de campos de encabezado, una línea en blanco y después el cuerpo del mensaje. Cada campo de encabezado consiste (lógicamente) en una sola línea de texto ASCII que contiene el nombre del campo, un signo de dos puntos y para la mayoría de los campos un valor. El RFC 822 original se diseñó hace varias décadas y no diferenciaba con claridad los campos de la envoltura de los campos del encabezado. Aunque se modificó para convertirse en el RFC 5322, no fue posible reconstruirlo por completo debido a su amplio uso. En el uso normal, el agente de usuario construye un mensaje y lo pasa al agente de transferencia de mensajes, que a su vez utiliza algunos de los campos del encabezado para construir la envoltura real, una mezcla algo anticuada de mensaje y envoltura.

En la figura 7-10 se listan los principales campos de encabezado relacionados con el transporte de mensajes. *To:* proporciona la dirección DNS del destinatario principal. También se permite tener varios destinatarios. *Cc*: proporciona las direcciones de los destinatarios secundarios. En términos de entrega, no hay distinción entre los destinatarios primarios y los secundarios. Es una diferencia totalmente psicológica que podría ser importante para las personas involucradas, pero para el sistema de correo no lo es. El término *Cc*: (Copia al carbón) es un poco obsoleto, ya que las computadoras no usan papel de carbón, pero está bien establecido. *Bcc*: (Copia al carbón oculta) es como el campo *Cc*:, sólo que esta línea se elimina de todas las copias que se envían a los destinatarios primarios y secundarios. Esta característica permite a las personas enviar copias a terceros sin que los destinatarios primarios y secundarios se enteren de ello.

Encabezado	Significado
To:	Dirección(es) de correo electrónico del (los) recipiente(s) primario(s).
Cc:	Dirección(es) de correo electrónico del (los) recipiente(s) secundario(s).
Bcc:	Dirección(es) de correo electrónico para las copias al carbón ocultas.
From:	Persona o personas que crearon el mensaje.
Sender:	Dirección de correo electrónico del emisor actual.
Received:	Línea que agrega cada agente de transferencia a lo largo de la ruta.
Return-path:	Se puede usar para identificar una ruta de vuelta al emisor.

Figura 7-10. Campos de encabezado del RFC 5322 relacionados con el transporte de mensajes.

Los siguientes dos campos, *From*: y *Sender*:, indican quién escribió y envió el mensaje, respectivamente. No necesitan ser iguales. Por ejemplo, un ejecutivo de negocios puede escribir un mensaje, pero su secretaria podría ser la que lo transmita. En este caso, el ejecutivo estaría listado en el campo *From*: y la secretaria en *Sender*:. *From*: es necesario, pero el campo *Sender*: se puede omitir si es igual a *From*:. Estos campos son necesarios en caso de que el mensaje no se pueda entregar y deba devolverse al remitente.

Se agrega una línea que contiene *Received*: por cada agente de transferencia de mensajes a lo largo de la trayectoria. La línea contiene la identidad del agente, la fecha y hora de recepción del mensaje, e información adicional que puede servir para depurar el sistema de enrutamiento.

El campo *Return-Path*: lo agrega el agente de transferencia de mensajes final y estaba destinado para indicar la manera de regresar al emisor. En teoría, esta información puede tomarse de todos los encabezados *Received*: (con excepción del nombre del buzón del emisor), pero pocas veces se llena de esta manera y por lo general contiene sólo la dirección del remitente.

Además de los campos de la figura 7-10, los mensajes del RFC 5322 también pueden contener una variedad de campos de encabezado usados por los agentes de usuario o los destinatarios. En la figura 7-11 se listan los más comunes. La mayoría de ellos se explican por sí solos, por lo que no los veremos con mucho detalle.

Encabezado	Significado
Date:	Fecha y hora de envío del mensaje.
Reply-To:	Dirección de correo electrónico a la que se deben enviar las respuestas.
Message-Id:	Número único para hacer referencia a este mensaje después.
In-Reply-To:	Identificador del mensaje al que éste responde.
References:	Otros identificadores de mensaje relevantes.
Keywords:	Palabras clave seleccionadas por el usuario.
Subject:	Resumen corto del mensaje para desplegar en una línea.

Figura 7-11. Algunos campos utilizados en el encabezado de mensaje del RFC 5322.

El campo *Reply-To*: a veces se usa cuando ni la persona que redactó el mensaje ni la que lo envió quieren ver la respuesta. Por ejemplo, suponga que un gerente de marketing escribe un mensaje de correo electrónico para informar a los clientes sobre un nuevo producto. El mensaje lo envía una secretaria, pero *Reply-To*: indica el jefe del Departamento de Ventas, quien puede contestar preguntas y surtir pedidos. Este campo también es útil cuando el emisor tiene dos cuentas de correo electrónico y desea que la respuesta llegue a su otra cuenta.

El campo *Message-Id*: es un número que se genera de manera automática y se utiliza para enlazar mensajes (por ejemplo, cuando se usa en *In-Reply-To*:), además de evitar la entrega de duplicados.

El documento RFC 5322 indica explícitamente que los usuarios pueden inventar encabezados opcionales para su propio uso privado. Por convención desde el RFC 822, estos encabezados empiezan con la cadena *X-*. Se garantiza que no habrá encabezados futuros que usen nombres que comiencen con *X-*, con el fin de evitar conflictos entre los encabezados oficiales y los privados. A veces algunos estudiantes universitarios, que se quieren pasar de listos, inventan campos como *X-Fruta-del-Día:* o *X-Enfermedad-de-la-Semana:*, que son legales, aunque no siempre son muy ilustrativos.

Después de los encabezados viene el cuerpo del mensaje. Los usuarios pueden poner aquí lo que les venga en gana. Algunos terminan sus mensajes con firmas complicadas, incluyendo citas de autoridades mayores y menores, pronunciamientos políticos y renuncias de responsabilidades de todo tipo (por ejemplo, la corporación ABC no es responsable de mis opiniones; de hecho, ni siquiera las puede entender).

MIME: Extensiones Multipropósito de Correo Internet

En los primeros días de ARPANET, el correo electrónico consistía exclusivamente en mensajes de texto escritos en inglés y expresados en ASCII. En tal entorno, el primer formato del RFC 822 hacía todo el trabajo: especificaba los encabezados, pero dejaba el contenido en manos del usuario. En la década de 1990,

el uso a nivel mundial de Internet y la demanda de enviar contenido más completo a través del sistema de correo significaba que este método ya no era adecuado. Los problemas incluían el envío y recepción de mensajes en idiomas con acentos (por ejemplo, español, francés y alemán), mensajes en alfabetos no latinos (por ejemplo, hebreo y ruso), mensajes en idiomas sin alfabetos (por ejemplo, chino y japonés) y mensajes que no contienen texto (por ejemplo, audio, imágenes o documentos y programas binarios).

La solución fue el desarrollo de **MIME** (**Extensiones Multipropósito de Correo Internet**, del inglés *Multipurpose Internet Mail Extensions*), que se usa ampliamente para los mensajes de correo que se envían a través de Internet, así como para describir contenido para otras aplicaciones, como la navegación web. MIME se describe en los RFC 2045-2047, 4288, 4289 y 2049.

La idea básica de MIME es seguir usando el formato RFC 822 (el precursor del RFC 5322 cuando se propuso MIME), pero agregar una estructura al cuerpo del mensaje y definir reglas de codificación para los mensajes que no son ASCII. Al no desviarse del RFC 822, se pudieron enviar mensajes MIME mediante el uso de los agentes de transferencia y protocolos de correo existentes (con base en el RFC 821 en ese entonces, y en el RFC 5321 ahora). Todo lo que había que cambiar eran los programas emisores y receptores; algo que los usuarios podían hacer por sí mismos.

MIME define cinco nuevos encabezados de mensaje, como se muestra en la figura 7-12. El primero de éstos simplemente indica al agente de usuario receptor del mensaje que está tratando con un mensaje MIME, así como la versión de MIME que usa. Se considera que cualquier mensaje que no contenga un encabezado *MIME-Version:* es un mensaje de texto plano en inglés (o por lo menos uno que utiliza caracteres ASCII), y se procesa como tal.

Encabezado	Significado
MIME-Version:	Identifica la versión MIME.
Content-Description:	Cadena legible por humanos que indica lo que contiene el mensaje.
Content-Id:	Identificador único.
Content-Transfer-Encoding:	Cómo se envuelve el mensaje para su transmisión.
Content-Type:	Tipo y formato del contenido.

Figura 7-12. Encabezados de mensaje agregados por MIME.

El encabezado *Content-Description*: es una cadena ASCII que indica lo que contiene el mensaje. Este encabezado es necesario para que el destinatario sepa si vale la pena decodificar y leer el mensaje. Si la cadena dice: "Foto del hámster de Bárbara" y la persona que recibe el mensaje no es un gran fanático de los hámsteres, lo más probable es que el mensaje se descarte en lugar de decodificarlo para obtener una foto a color de alta definición.

El encabezado *Content-Id*: identifica el contenido. Usa el mismo formato que el encabezado *Message-Id*: estándar.

El encabezado *Content-Transfer-Encoding*: indica la manera en que está envuelto el mensaje para transmitirlo a través de la red. Un problema clave al momento de desarrollar MIME era que los protocolos de transferencia de correo (SMTP) esperaban mensajes ASCII en los que ninguna línea era mayor a 1000 caracteres. Los caracteres ASCII usan 7 bits de cada byte de 8 bits. Los datos binarios como programas ejecutables e imágenes usan los 8 bits de cada byte, al igual que los conjuntos de caracteres extendidos. No había garantía de que estos datos se transfirieran en forma segura. Por ende, se requería algún método

para transportar datos binarios que hiciera ver a estos datos como un mensaje de correo ASCII regular. Las extensiones al SMTP desde el desarrollo de MIME permiten transferir datos binarios de 8 bits, aun cuando en la actualidad los datos binarios no siempre pasan correctamente a través del sistema de correo si no están codificados.

MIME cuenta con cinco esquemas de codificación de transferencia, además de un escape para nuevos esquemas (por si acaso). El esquema más simple es el de los mensajes simples de texto ASCII. Los caracteres ASCII usan 7 bits y el protocolo de correo electrónico los puede transportar directamente, siempre y cuando ninguna línea sea mayor a 1000 caracteres.

El siguiente esquema más sencillo muestra lo mismo, pero utiliza caracteres de 8 bits; es decir, se permiten todos los valores de 0 a 255, inclusive. Los mensajes que usan la codificación de 8 bits se tienen que adherir también a la longitud máxima de línea estándar.

Después están los mensajes que usan una verdadera codificación binaria. Éstos son archivos binarios arbitrarios que no sólo usan los 8 bits, sino que además no se adhieren al límite de línea de 1000 caracteres. Los programas ejecutables se encuentran en esta categoría. Hoy en día los servidores de correo pueden negociar el envío de datos en codificación binaria (o de 8 bits), y cambian de vuelta a ASCII si el emisor o el receptor no soportan la extensión.

La codificación ASCII de datos binarios se conoce como **codificación base64**. En este esquema se dividen grupos de 24 bits en cuatro unidades de 6 bits, y cada unidad se envía como un carácter ASCII legal. La codificación es “A” para 0, “B” para 1, etc., seguidas por las 26 letras minúsculas, los 10 dígitos y, por último, + y / para el 62 y 63, respectivamente. Las secuencias == e = se usan para indicar que el último grupo contenía sólo 8 o 16 bits, respectivamente. Los retornos de carro y avances de línea se ignoran, por lo que pueden introducirse a voluntad para mantener la longitud de línea lo suficientemente corta. Podemos enviar un texto binario arbitrario en forma segura mediante este esquema, aunque de manera ineficiente. Esta codificación fue muy popular antes de que se extendiera la implementación de los servidores de correo con capacidad para datos binarios. Hoy en día se sigue utilizando con regularidad.

En el caso de mensajes que son casi completamente ASCII, pero con algunos caracteres no ASCII, la codificación base 64 es algo ineficiente. En su lugar se usa una codificación conocida como **codificación entrecomillada imprimible**, la cual consiste simplemente en código ASCII de 7 bits, con todos los caracteres por encima de 127 codificados como un signo de igual seguido del valor del carácter en dos dígitos hexadecimales. También se codifican los caracteres de control, algunos signos de puntuación y símbolos matemáticos, al igual que los espacios finales.

Por último, cuando existen razones válidas para no utilizar alguno de estos esquemas, es posible especificar una codificación definida por el usuario en el encabezado *Content-Transfer-Encoding*:.

El último encabezado mostrado en la figura 7-12 es en realidad el más interesante. Especifica la naturaleza del cuerpo del mensaje y ha tenido un impacto mucho más allá del correo electrónico. Por ejemplo, el contenido que se descarga de la web se etiqueta mediante tipos MIME para que el explorador web sepa cómo presentarlo. Lo mismo ocurre con el contenido que se envía a través de medios de flujo continuo y transportes en tiempo real, como la tecnología de voz sobre IP.

En un principio se definieron siete tipos MIME en el RFC 1521. Cada uno tiene uno o más subtipos. El tipo y el subtipo se separan mediante una diagonal, como en “Content-Type: video/mpeg”. Desde entonces se han agregado cientos de subtipos, junto con otro tipo. Las entradas adicionales se agregan todo el tiempo a medida que se desarrollan nuevos tipos de contenido. La IANA se encarga de mantener en línea la lista de tipos y subtipos asignados, la cual se encuentra en *www.iana.org/assignmens/media-types*.

En la figura 7-13 se muestran los tipos, junto con ejemplos de los subtipos de uso común. Daremos un breve repaso de ellos, empezando con *text*. La combinación *text/plain* es para mensajes ordinarios que se pueden visualizar así como se reciben, sin codificación ni ningún procesamiento posterior. Esta opción permite el transporte de mensajes ordinarios en MIME con sólo unos cuantos encabezados adicionales.

Tipo	Subtipos de ejemplo	Descripción
text	plain, html, xml, css	Texto en diversos formatos.
image	gif, jpeg, tiff	Imágenes.
audio	basic, mpeg, mp4	Sonidos.
video	mpeg, mp4, quicktime	Películas.
model	vrml	Modelo 3D.
application	octet-stream, pdf, javascript, zip	Datos producidos por aplicaciones.
message	http, rfc822	Mensaje encapsulado.
multipart	mixed, alternative, parallel, digest	Combinación de múltiples tipos.

Figura 7-13. Tipos de contenido MIME y subtipos de ejemplo.

El subtipo *text/html* se agregó cuando la web se hizo popular (en el RFC 2854), para enviar páginas web en el correo del RFC 822. En el RFC 3023 se define un subtipo para el Lenguaje de Marcado eXtensible: *text/xml*. Los documentos XML han proliferado con el desarrollo de la web. En la sección 7.3 estudiaremos los lenguajes HTML y XML.

El siguiente tipo MIME es *image*, que se usa para transmitir imágenes fijas. Hoy día se usan muchos formatos para almacenar y transmitir imágenes, tanto con compresión como sin ella. Varios de éstos, incluyendo GIF, JPEG y TIFF, están integrados en la mayoría de los navegadores. También existen muchos otros formatos y los subtipos correspondientes.

Los tipos *audio* y *video* son para sonido e imágenes en movimiento, respectivamente. Observe que *video* puede incluir sólo la información visual, no el sonido. Si se debe transmitir una película con sonido, tal vez sea necesario transmitir las partes de video y de audio por separado, dependiendo del sistema de codificación que se utilice. El primer formato de video definido fue el diseñado por el grupo denominado modestamente como MPEG (Grupo de Expertos en Imágenes en Movimiento, del inglés *Moving Picture Experts Group*), pero desde entonces se han agregado otros. Además de *audio/basic*, se agregó un nuevo tipo de audio, *audio/mpeg*, en el RFC 3003 para permitir que las personas pudieran enviar archivos de audio MP3. Los tipos *video/mp4* y *audio/mp4* son para datos de audio y video que se almacenan en el formato MPEG4 más reciente.

El tipo *model* se agregó después de los otros tipos de contenido. Su objetivo es describir los datos de modelos 3D. Sin embargo, a la fecha no se ha utilizado ampliamente.

El tipo *application* es un tipo general para los formatos que no están cubiertos por uno de los otros tipos y requieren una aplicación para interpretar los datos. Hemos listado los subtipos *pdf*, *javascript* y *zip* como ejemplos para los documentos PDF, programas de JavaScript y archivos Zip, respectivamente. Los agentes de usuario que reciben este contenido utilizan una biblioteca de terceros o un programa externo para mostrarlo; la pantalla puede tener o no la apariencia de estar integrada al agente de usuario.

Mediante el uso de los tipos MIME, los agentes de usuario obtienen la capacidad de extenderse para manejar nuevos tipos de contenido de aplicaciones, a medida que se van desarrollando. Esto es un beneficio importante. Por otra parte, muchas de las nuevas formas de contenido se ejecutan mediante aplicaciones, lo cual presenta ciertos peligros. Sin duda, ejecutar un programa arbitrario que llega del sistema de correo de parte de unos "amigos" representa un riesgo de seguridad. El programa puede realizar todo tipo de daños a las partes de la computadora a la que tiene acceso, en especial si puede leer y escribir archivos, además de usar la Red. Algo no tan obvio es el hecho de que los formatos de los documentos pueden presentar los mismos riesgos. Esto se debe a que los formatos como PDF son lenguajes de programación

completos. Aunque se interpretan y están restringidos en cuanto al alcance, con frecuencia los errores en el intérprete permiten a los documentos deshonestos escapar a esas restricciones.

Además de estos ejemplos, hay muchos otros subtipos de aplicaciones debido a que existen muchas aplicaciones más. Como alternativa a usar cuando no se conoce ningún otro subtipo más adecuado, el subtipo *octet-stream* denota una secuencia de bytes no interpretados. Al recibir un flujo de este tipo, es probable que un agente de usuario lo muestre en pantalla y sugiera al usuario que lo copie en un archivo. A partir de entonces, el procesamiento posterior es responsabilidad del usuario, quien se supone debe saber qué tipo de contenido es.

Los dos últimos tipos son útiles para redactar y manipular los mensajes mismos. El tipo *message* permite que un mensaje esté encapsulado por completo dentro de otro. Este esquema es útil para reenviar, por ejemplo, correo electrónico. Cuando se encapsula un mensaje RFC 822 completo en un mensaje exterior, es necesario usar el subtipo *rfc822*. De manera similar, es común que los documentos de HTML estén encapsulados. Y el subtipo *partial* hace posible dividir un mensaje encapsulado en varias piezas y enviarlas por separado (por ejemplo, si el mensaje encapsulado es demasiado extenso). Los parámetros hacen posible reensamblar todas las partes en el destino y en el orden correcto.

Por último, el tipo *multipart* permite que un mensaje contenga más de una parte, con el comienzo y el fin de cada parte delimitados. El subtipo *mixed* permite que cada parte sea de un tipo diferente, sin ninguna estructura adicional impuesta. Muchos programas de correo electrónico permiten que el usuario agregue uno o más archivos adjuntos a un mensaje de texto. Estos archivos adjuntos se envían mediante el tipo *multipart*.

A diferencia de *mixed*, el subtipo *alternative* permite que el mismo mensaje se incluya varias veces, pero se exprese en dos o más medios diferentes. Por ejemplo, un mensaje se podría enviar en ASCII común, en HTML y en PDF. Un agente de usuario diseñado de manera apropiada que reciba uno de tales mensajes lo mostraría de acuerdo con las preferencias del usuario. Es probable que PDF fuera la primera opción, si estuviera disponible. La segunda opción sería HTML. Si ninguna de estas dos opciones estuvieran disponibles, entonces se mostraría el texto ASCII común. Las partes deben ordenarse de la más sencilla a la más compleja para ayudar a que los receptores con agentes de usuario pre-MIME puedan encontrarle algún sentido al mensaje (por ejemplo, incluso un usuario pre-MIME puede leer texto ASCII común).

El subtipo *alternative* también se puede usar para varios lenguajes. En este contexto, puede pensarse en el software Rosetta Stone como uno de los primeros mensajes *multipart/alternative*.

De los otros dos subtipos de ejemplo, el subtipo *parallel* se utiliza cuando todas las partes se tienen que "ver" simultáneamente. Por ejemplo, a menudo las películas tienen un canal de audio y uno de video; son más efectivas si estos dos canales se reproducen en paralelo, en vez de hacerlo en forma consecutiva. El subtipo *digest* se utiliza cuando varios mensajes se empaquetan juntos en un solo mensaje compuesto. Por ejemplo, ciertos grupos de discusión en Internet recolectan mensajes de los suscriptores y después los envían al grupo en forma periódica como un solo mensaje *multipart/digest*.

Como ejemplo de la forma en que se pueden usar los tipos MIME para los mensajes de correo electrónico, en la figura 7-14 se muestra un mensaje multimedia. Aquí se transmite una felicitación de cumpleaños en formas alternativas, como HTML y como un archivo de audio. Si suponemos que el receptor tiene capacidad de audio, su agente de usuario reproducirá el archivo de sonido. En este ejemplo, el sonido se transporta por referencia como un subtipo *message/external-body*, por lo que primero el agente de usuario debe obtener el archivo de sonido *cumple.snd* mediante el uso de FTP. Si el agente de usuario no tiene capacidad de audio, se muestran las letras en pantalla en absoluto silencio. Las dos partes están delimitadas por dos guiones seguidos de una cadena (generada por software) especificada en el parámetro *boundary*.

Cabe mencionar que el encabezado *Content-Type* aparece en tres lugares en este ejemplo. En el nivel superior, indica que el mensaje tiene varias partes. En cada parte, indica el tipo y el subtipo de esa parte.

Por último, en el cuerpo de la segunda parte, es necesario para indicarle al agente de usuario la clase de archivo externo que debe conseguir. Para indicar esta pequeña diferencia de uso, hemos usado letras en minúscula, aunque los encabezados no hacen distinciones entre mayúsculas y minúsculas. De la misma manera, se requiere el encabezado *Content-Transfer-Encoding* para cualquier cuerpo externo que no esté codificado como ASCII de 7 bits.

7.2.4 Transferencia de mensajes

Ahora que hemos descrito a los agentes de usuario y los mensajes de correo, estamos listos para ver cómo es que los agentes de transferencia transmiten los mensajes del remitente al destinatario. La transferencia de correo se realiza mediante el protocolo SMTP.

La manera más sencilla de mover mensajes es establecer una conexión de transporte de la máquina de origen a la de destino y después sólo transferir el mensaje. Así es como trabajaba SMTP en un principio. Sin embargo, con el paso de los años se han distinguido dos usos de SMTP. El primer uso es el envío de correo, el paso 1 en la arquitectura del correo electrónico de la figura 7-7. Éste es el medio por el cual los agentes de usuario envían mensajes al sistema de correo para su entrega. El segundo uso es transferir mensajes entre los agentes de transferencia (paso 2 de la figura 7-7). Esta secuencia entrega correo desde el emisor hasta el agente de transferencia de mensajes receptor en un solo salto. La entrega final se realiza mediante distintos protocolos, que describiremos en la siguiente sección.

En esta sección describiremos los fundamentos del protocolo SMTP y su mecanismo de extensión. Después veremos cómo se usa de manera distinta para enviar correo y transferir mensajes.

```
From: alice@cs.washington.edu
To: bob@ee.uwa.edu.au
MIME-Version: 1.0
Message-Id: <0704760941.AA00747@cs.washington.edu>
Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm
Subject: La Tierra da vuelta al Sol un número entero de veces

Éste es el preámbulo. El agente de usuario lo ignora. Tenga un bonito día.

--qwertyuiopasdfghjklzxcvbnm
Content-Type: text/html

<p>Feliz cumpleaños a ti<br>
Feliz cumpleaños a ti<br>
Feliz cumpleaños, querido<b> Bob </b><br>
Feliz cumpleaños a ti</p>

--qwertyuiopasdfghjklzxcvbnm
Content-Type: message/external-body;
        access-type=”anon-ftp”;
        site=”bicycle.cs.washington.edu”;
        directory=”pub”;
        name=”cumple.snd”

content-type: audio/basic
content-transfer-encoding: base64
--qwertyuiopasdfghjklzxcvbnm—
```

Figura 7-14. Un mensaje multipartita que contiene alternativas de HTML y audio.

SMTP (Protocolo Simple de Transferencia de Correo) y sus extensiones

En Internet, el correo electrónico se entrega al hacer que la computadora emisora establezca una conexión TCP con el puerto 25 de la computadora receptora. En este puerto escucha un servidor de correo que habla **SMTP** (**Protocolo Simple de Transferencia de Correo**, del inglés *Simple Mail Transfer Protocol*). Este servidor acepta conexiones entrantes sujetas a ciertas verificaciones de seguridad, y además acepta mensajes para su entrega. Si no se puede entregar un mensaje, se devuelve al emisor un informe de error que contiene la primera parte del mensaje que no se pudo entregar.

SMTP es un protocolo ASCII simple. Esto no es una debilidad, sino una característica. Al usar texto ASCII se facilita el proceso de desarrollar, probar y depurar los protocolos. Se pueden probar al enviar comandos en forma manual, y los registros de los mensajes son fáciles de leer. La mayoría de los protocolos de Internet a nivel de aplicación funcionan ahora de esta manera (por ejemplo, HTTP).

Vamos a recorrer una transferencia de un mensaje simple entre servidores de correo para entregar un mensaje. Después de establecer la conexión TCP con el puerto 25, la máquina emisora, que opera como cliente, espera a que la máquina receptora, que opera como servidor, transmita primero. El servidor empieza por enviar una línea de texto en la que proporciona su identidad e indica si está preparado o no para recibir correo. Si no lo está, el cliente libera la conexión y lo intenta después.

Si el servidor está dispuesto a aceptar correo electrónico, el cliente anuncia de quién proviene el mensaje, y a quién está dirigido. Si existe el destinatario en el destino, el servidor da al cliente permiso para enviar el mensaje. A continuación el cliente envía el mensaje y el servidor confirma su recepción. No se requieren sumas de verificación porque TCP proporciona un flujo de bytes confiable. Si hay más correo electrónico, se envía ahora. Una vez que se ha intercambiado todo el correo electrónico en ambas direcciones, se libera la conexión. En la figura 7-15 se muestra un diálogo simple de ejemplo para enviar el mensaje de la figura 7-14, incluyendo los códigos numéricos que utiliza SMTP. Las líneas enviadas por el cliente (es decir, el emisor) se marcan con *C:*. Aquellas que envía el servidor (es decir el receptor) se marcan con *S*:.

El primer comando del cliente es *HELO*. De las diversas abreviaturas de cuatro caracteres de *HELLO*, ésta tiene numerosas ventajas sobre su mayor competidora. La razón por la que los comandos tienen que ser de cuatro caracteres se ha perdido en las brumas del tiempo.

En la figura 7-15 el mensaje se envía a un solo destinatario, por lo que se usa un solo comando *RCPT*. Se permiten tales comandos para enviar un solo mensaje a múltiples destinatarios. Se confirma la recepción de cada uno o se rechaza de manera individual. Incluso si se rechazan algunos destinatarios (porque no existen en el destino), el mensaje se puede enviar a los demás.

Por último, aunque la sintaxis de los comandos de cuatro caracteres del cliente se especifica con rigidez, la sintaxis de las respuestas es menos rígida. Sólo cuenta el código numérico. Cada implementación puede poner la cadena que desee después del código.

El protocolo SMTP básico funciona bien, pero está limitado en varios aspectos. No incluye autentificación. Esto significa que el comando *FROM* en el ejemplo podría proporcionar cualquier dirección de emisor que desee. Esto es bastante útil para enviar spam. Otra limitación es que el SMTP transfiere mensajes ASCII, no datos binarios. Esto explica por qué era necesaria la codificación de transferencia de contenido MIME base64. Sin embargo, con esa codificación la transmisión de correo usa el ancho de banda en forma ineficiente, lo cual representa un problema para los mensajes más grandes. Una tercera limitación es que el SMTP envía mensajes sin ningún tipo de cifrado para proveer una medida de privacidad para protegerse contra los entrometidos.

Para lidiar con éstos y otros problemas relacionados con el procesamiento de los mensajes, se revisó el protocolo SMTP para idear un mecanismo de extensión. Este mecanismo es una parte obligatoria del estándar RFC 5321. El uso de SMTP con extensiones se denomina **ESMTP** (**SMPT Extendido**, del inglés *Extended SMTP*).

```
          S: 220 ee.uwa.edu.au SMTP service ready
C: HELO abcd.com
          S: 250 cs.washington.edu says hello to ee.uwa.edu.au
C: MAIL FROM:<alice@cs.washington.edu>
          S: 250 sender ok
C: RCPT TO:<bob@ee.uwa.edu.au>
          S: 250 recipient ok
C: DATA
          S: 354 Send mail; end with "." on a line by itself
C: From: alice@cs.washington.edu
C: To: bob@ ee.uwa.edu.au
C: MIME-Version: 1.0
C: Message-Id: <0704760941.AA00747@ee.uwa.edu.au >
C: Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm
C: Subject: La Tierra da vuelta al Sol un número entero de veces
C:
C: Éste es el preámbulo. El agente de usuario lo ignora. Tenga un bonito día.
C:
C: --qwertyuiopasdfghjklzxcvbnm
C: Content-Type: text/html
C:
C: <p>Feliz cumpleaños a ti
C: Feliz cumpleaños a ti
C: Feliz cumpleaños, querido <bold> Bob </bold>
C: Feliz cumpleaños a ti
C:
C: --qwertyuiopasdfghjklzxcvbnm
C: Content-Type: message/external-body;
C:        access-type="anon-ftp";
C:        site="bicycle.cs.washington.edu";
C:        directory="pub";
C:        name="cumple.snd"
C:
C: content-type: audio/basic
C: content-transfer-encoding: base64
C: --qwertyuiopasdfghjklzxcvbnm
C: .
          S: 250 message accepted
C: QUIT
          S: 221 ee.uwa.edu.au closing connection
```

Figura 7-15. Envío de un mensaje de *alice@cs.washington.edu* a *bob@ee.uwa.edu.au*.

Los clientes que deseen usar una extensión envían al principio un mensaje *EHLO* en vez de *HELO*. Si el saludo se rechaza, esto indica que es un servidor SMTP normal, y el cliente debe proceder de la manera normal. Si se acepta el *EHLO*, el servidor responde con las extensiones que soporta. A continuación el cliente puede usar cualquiera de estas extensiones. En la figura 7-16 se muestran varias extensiones comunes. La figura muestra cómo se utiliza la palabra clave en el mecanismo de extensión, junto con una descripción de la nueva funcionalidad. No entraremos en más detalles respecto a las extensiones.

Para obtener una mejor idea de la forma en que trabajan tanto SMTP como algunos de los demás protocolos descritos en este capítulo, hay que probarlos. En todos los casos, primero vaya a una máquina conectada a Internet. En un sistema UNIX (o Linux), en una línea de comando, escriba:

```
telnet mail.isp.com 25
```

Palabra clave	Descripción
AUTH	Autentificación del cliente.
BINARYMIME	El servidor acepta mensajes binarios.
CHUNKING	El servidor acepta mensajes extensos en partes.
SIZE	Verifica el tamaño del mensaje antes de intentar enviarlo.
STARTTLS	Cambiar a transporte seguro (TLS; vea el capítulo 8).
UTF8SMTP	Direcciones internacionalizadas.

Figura 7-16. Algunas extensiones SMTP.

y sustituya el nombre DNS de su servidor de correo ISP por *mail.isp.com*. En un sistema Windows XP, haga clic en Inicio | Ejecutar (Start | Run) y después escriba el comando en el cuadro de diálogo. En un equipo con Vista o Windows 7, tal vez primero tenga que instalar el programa telnet (o su equivalente) y después iniciarlo por su cuenta. Este comando establecerá una conexión telnet (es decir, TCP) con el puerto 25 de esa máquina. El puerto 25 es el puerto SMTP; en la figura 6-34 puede consultar los puertos para otros protocolos comunes. Es probable que obtenga una respuesta parecida a lo siguiente:

```
Trying 192.30.200.66...
Connected to mail.isp.com
Escape character is '^]'.
220 mail.isp.com Smail #74 ready at Thu, 25 Sept 2002 13:26 +0200
```

Las primeras tres líneas son de telnet y le indican lo que están haciendo. La última línea es del servidor SMTP de la máquina remota, que anuncia su disponibilidad para hablar con usted y aceptar correo electrónico. Para ver cuáles comandos acepta, escriba:

```
HELP
```

De aquí en adelante, es posible una secuencia de comandos como la que se muestra en la figura 7-16, si el servidor está dispuesto a aceptar su correo.

Envío de correo

En un principio los agentes de usuario se ejecutaban en la misma computadora que el agente de transferencia de mensajes emisor. En esta configuración, todo lo que se requiere para enviar un mensaje es que el agente de usuario hable con el servidor de correo local, mediante el diálogo que describimos antes. Sin embargo, esta configuración ya no es el caso común.

Con frecuencia, los agentes de usuario se ejecutan en laptops, computadoras domésticas y teléfonos móviles. No siempre están conectados a Internet. Los agentes de transferencia de correo se ejecutan en los servidores del ISP y de las empresas. Siempre están conectados a Internet. Esta diferencia significa que un agente de usuario en Boston tal vez necesite ponerse en contacto con su servidor de correo regular en Seattle para enviar un mensaje, debido a que el usuario se encuentra de viaje.

Por sí sola, esta comunicación remota no representa un problema. Es exactamente lo que los protocolos TCP/IP están diseñados para soportar. Sin embargo, un ISP o empresa por lo general no desean que un usuario remoto pueda enviar mensajes a su servidor de correo para entregarlos en cualquier otra parte. El ISP o empresa no ejecuta el servicio como si fuera público. Además, este tipo de **retransmisor** (*relay*)

de correo abierto atrae a los *spammers*. Esto se debe a que provee una forma de "lavar" el emisor original y de esta forma hacer que el mensaje sea más difícil de identificar como spam.

Con base en estas consideraciones, por lo general el SMTP se utiliza para enviar correo con la extensión *AUTH*. Esta extensión permite al servidor verificar las credenciales (nombre de usuario y contraseña) del cliente, para confirmar que el servidor le pueda proporcionar servicio de correo.

Existen diversas diferencias más en la forma en que se utiliza SMTP para enviar correo. Por ejemplo, es preferible utilizar el puerto 587 en vez del puerto 25 y el servidor SMTP puede tanto verificar como corregir el formato de los mensajes que envía el agente de usuario. Para obtener más información sobre el uso restringido de SMTP para enviar correo, consulte el RFC 4409.

Transferencia de mensajes

Una vez que el agente de transferencia de mensajes recibe un mensaje del agente de usuario, lo entrega al agente de transferencia de correo receptor mediante SMTP. Para ello, el emisor usa la dirección de destino. Considere el mensaje de la figura 7-15, dirigido a *bob@ee.uwa.edu.au*. ¿A qué servidor de correo se debe entregar el mensaje?

Para determinar el correo servidor de correcto que se debe contactar, es necesario consultar al DNS. En la sección anterior describimos cómo el DNS contiene varios tipos de registros, incluyendo el *MX*, o intercambiador de correo. En este caso se realiza una consulta DNS sobre los registros *MX* del dominio *ee.uwa.edu.au*. Esta consulta devuelve una lista ordenada de los nombres y direcciones IP de uno o más servidores de correo.

A continuación, el agente de transferencia de correo emisor realiza una conexión TCP en el puerto 25 a la dirección IP del servidor de correo para comunicarse con el agente de transferencia de correo receptor, y usa SMTP para transmitir el mensaje. A su vez, el agente de transferencia de correo receptor colocará el correo para el usuario *bob* en el buzón correcto para que Bob lo lea después. Tal vez en este paso de entrega local sea necesario mover el mensaje entre computadoras, si hay una gran infraestructura de correo.

Con este proceso de entrega, el correo viaja del agente de transferencia de correo inicial al agente final en un solo salto. No hay servidores intermedios en la etapa de transferencia del mensaje. Sin embargo, es posible que este proceso de entrega ocurra varias veces. Un ejemplo que ya describimos antes es cuando un agente de transferencia de mensajes implementa una lista de correo. En este caso, se recibe un mensaje para la lista. Después, este mensaje se envía a cada una de las direcciones de la lista de los miembros individuales.

Como otro ejemplo de transmisión, tal vez Bob se haya graduado del MIT y también se pueda llegar a él por medio de la dirección *bob@alum.mit.edu*. En vez de leer el correo de varias cuentas, Bob puede hacer que el correo enviado a esta dirección se reenvíe a *bob@ee.uwa.edu*. En este caso, el correo que se envíe a *bob@alum.mit.edu* pasará por dos entregas. Primero se enviará al servidor de correo para *alum.mit.edu*. Después se enviará al servidor de correo para *ee.uwa.edu.au*. Cada uno de estos tramos es una entrega completa y separada, en lo que respecta a los agentes de transferencia de correo.

Otro aspecto a considerar en la actualidad es el spam. Nueve de cada 10 mensajes que se envían actualmente son spam (McAfee, 2010). Pocas personas desean más spam, pero es difícil evitar debido a que se hace pasar por correo regular. Antes de aceptar un mensaje, hay que llevar a cabo verificaciones adicionales para reducir las oportunidades para el spam. El mensaje para Bob se envió desde *alice@cs.washington.edu*. El agente de transferencia de correo receptor puede buscar el agente de transferencia de correo emisor en el DNS. Esto le permite verificar que la dirección IP del otro extremo de la conexión TCP coincida con el nombre DNS. En forma más general, el agente receptor puede buscar el dominio emisor en el DNS para ver si tiene una política de envío de correo. Esta información se proporciona con frecuencia en los registros *TXT* y *SPF*. Podría indicar que es posible realizar otras verificaciones. Por ejemplo,

el correo enviado desde *cs.washington.edu* puede enviarse siempre desde el host *june.cs.washington.edu*. Si el agente de transferencia de correo emisor no es *june*, significa que hay un problema.

Si falla alguna de estas verificaciones, es probable que el correo se esté alterando con una dirección de envío falsa. En este caso, se desecha. Sin embargo, pasar estas verificaciones no significa que el correo no sea spam. Estas verificaciones sólo aseguran que el correo parece provenir de la región de la red de la que pretende provenir. La idea es que los spammers se vean obligados a usar la dirección de envío correcta cuando envíen el correo. Esto facilita el reconocimiento y la eliminación del spam cuando no se desea.

7.2.5 Entrega final

Casi entregamos nuestro mensaje de correo. Ya llegó al buzón de correo de Bob. Todo lo que resta es transferir una copia del mensaje al agente de usuario de Bob para visualizarlo en pantalla. Éste es el paso 3 en la arquitectura de la figura 7-7. Esta tarea era sencilla en los primeros días de Internet, cuando el agente de usuario y el agente de transferencia de correo se ejecutaban en la misma máquina como procesos diferentes. El agente de transferencia de correo simplemente escribía nuevos mensajes al final del archivo del buzón de correo y el agente de usuario sólo revisaba el archivo del buzón de correo para ver si había correo nuevo.

En la actualidad es probable que el agente de usuario en una PC, laptop o equipo móvil se encuentre en una máquina distinta al servidor de correo del ISP o de la empresa. Los usuarios desean acceder a su correo en forma remota, desde cualquier lugar en donde se encuentren. Desean acceder al correo electrónico desde su trabajo, su hogar, sus laptops al viajar por cuestión de negocios, y desde los cibercafés cuando se suponga estén de vacaciones. También desean trabajar sin estar en línea, y después volverse a conectar para recibir el correo entrante y enviar el correo saliente. Lo que es más, cada usuario puede llegar a ejecutar varios agentes de usuario, dependiendo de la computadora que tenga disponible en un momento dado. Incluso puede haber varios agentes de usuario en ejecución al mismo tiempo.

En esta configuración, la tarea del agente de usuario es presentar una vista del contenido del buzón de correo y permitir que éste se manipule en forma remota. Se pueden usar varios protocolos distintos para este fin, pero SMTP no es uno de ellos. SMTP es un protocolo basado en *push* (empuje). Toma un mensaje y se conecta con un servidor remoto para transferirlo. La entrega final no se puede lograr de esta manera, debido a que el buzón de correo debe seguir almacenado en el agente de transferencia de correo y tal vez no esté conectado a Internet en el momento en que el SMTP trate de transmitir los mensajes.

IMAP: Protocolo de Acceso a Mensajes de Internet

Uno de los principales protocolos que se utilizan para la entrega final es **IMAP** (**Protocolo de Acceso a Mensajes de Internet**, del inglés *Internet Message Access Protocol*). La versión 4 del protocolo se define en el RFC 3501. Para usar IMAP, el servidor de correo ejecuta un servidor IMAP que escucha en el puerto 143. El agente de usuario ejecuta un cliente IMAP. El cliente se conecta al servidor y empieza a emitir comandos de los que se listan en la figura 7-17.

Primero, el cliente iniciará un transporte seguro si se debe usar uno (para mantener los mensajes y comandos confidenciales), y después iniciará sesión o se autentificará de alguna otra forma con el servidor. Una vez conectado, hay muchos comandos para listar carpetas y mensajes, obtener mensajes o incluso partes de ellos, marcar mensajes con banderas para eliminarlos después, y organizar los mensajes en carpetas. Para evitar confusión, tenga en cuenta que aquí usamos el término "carpeta" para que sea consistente con el resto del material en esta sección, en donde un usuario tiene un solo buzón de correo

Comando	Descripción
CAPABILITY	Lista las capacidades del servidor.
STARTTLS	Inicia transporte seguro (TLS; vea el capítulo 8).
LOGIN	Inicia sesión en el servidor.
AUTHENTICATE	Inicia sesión con otro método.
SELECT	Selecciona una carpeta.
EXAMINE	Selecciona una carpeta de sólo lectura.
CREATE	Crea una carpeta.
DELETE	Elimina una carpeta.
RENAME	Cambia el nombre a una carpeta.
SUBSCRIBE	Agrega carpeta al conjunto activo.
UNSUBSCRIBE	Elimina carpeta del conjunto activo.
LIST	Lista las carpetas disponibles.
LSUB	Lista las carpetas activas.
STATUS	Obtiene el estado de una carpeta.
APPEND	Agrega un mensaje a una carpeta.
CHECK	Obtiene un punto de verificación de una carpeta.
FETCH	Obtiene mensajes de una carpeta.
SEARCH	Busca mensajes en una carpeta.
STORE	Altera las banderas de los mensajes.
COPY	Crea una copia de un mensaje en una carpeta.
EXPUNGE	Elimina los mensajes marcados para eliminación.
UID	Emite comandos mediante el uso de identificadores únicos.
NOOP	No hace nada.
CLOSE	Elimina los mensajes marcados con banderas y cierra la carpeta.
LOGOUT	Cierra la sesión y la conexión.

Figura 7-17. Comandos IMAP (versión 4).

compuesto de varias carpetas. Sin embargo, en la especificación IMAP se utiliza el término *buzón de correo*. Así, un usuario tiene muchos buzones de correo IMAP, cada uno de los cuales se presenta comúnmente al usuario como una carpeta.

IMAP tiene también muchas otras características, como la habilidad de direccionar el correo no con base en el número de mensaje, sino mediante el uso de atributos (por ejemplo, dame el primer mensaje de Alice). Se pueden realizar búsquedas en el servidor para encontrar los mensajes que cumplan con ciertos criterios, de modo que el cliente recupere sólo esos mensajes.

IMAP es una mejora de uno de los primeros protocolos de entrega final, **POP3** (**Protocolo de Oficina de Correos, versión 3**; del inglés *Post Office Protocol, version 3*), el cual se especifica en el RFC

1939. POP3 es un protocolo más simple, pero soporta menos características y es menos seguro en el uso común. Por lo general el correo se descarga a la computadora del agente de usuario, en vez de permanecer en el servidor de correo. Esto facilita la vida al servidor, pero es más difícil para el usuario. No es fácil leer el correo en varias computadoras; además, si la computadora del agente de usuario se descompone, cabe la probabilidad de perder todo el correo electrónico de manera permanente. Sin embargo, el protocolo POP3 se sigue utilizando en muchas partes.

También se pueden usar protocolos propietarios, debido a que el protocolo se ejecuta entre un servidor de correo y un agente de usuario que pueden ser suministrados por la misma empresa. Microsoft Exchange es un sistema de correo con un protocolo propietario.

Correo web

Una alternativa cada vez más popular a IMAP y SMTP para proveer servicio de correo electrónico es utilizar la web como interfaz para enviar y recibir correo. Los sistemas de *correo web* que se utilizan ampliamente son Google Gmail, Microsoft Hotmail y Yahoo! Mail. El correo web es un ejemplo de software (en este caso, un agente de usuario de correo) que se proporciona como servicio mediante el uso de la web.

En esta arquitectura, el proveedor ejecuta los servidores de correo de la manera usual para aceptar los mensajes para los usuarios, con SMTP en el puerto 25. Sin embargo, el agente de usuario es distinto. En vez de ser un programa independiente, es una interfaz de usuario que se provee a través de páginas web. Esto significa que los usuarios pueden usar el navegador que deseen para acceder a su correo y enviar nuevos mensajes.

Aún no hemos estudiado la web, pero a continuación le daremos una breve descripción que puede utilizar como referencia en lo futuro. Cuando el usuario visita la página web de correo electrónico del proveedor, aparece un formulario en el que se le pide que introduzca un nombre de inicio de sesión y una contraseña. El nombre de inicio de sesión y la contraseña se envían al servidor, quien procede a validarlas. Si el inicio de sesión es exitoso, el servidor encuentra el buzón de correo del usuario y crea una página web en la que lista el contenido del buzón de correo al instante. Después, la página web se envía al explorador para que la muestre en pantalla.

Es posible hacer clic en muchos de los elementos en la página que muestra el buzón de correo, de modo que los mensajes se puedan leer, eliminar, etc. Para que la interfaz sea receptiva, es común que las páginas web incluyan programas de JavaScript. Estos programas se ejecutan en forma local en el cliente, en respuesta a los eventos locales (por ejemplo, los clics del ratón); además pueden descargar y enviar mensajes en segundo plano, para preparar el siguiente mensaje a visualizar o un nuevo mensaje para su envío. En este modelo, el envío de correo ocurre mediante el uso de los protocolos web normales, en donde se publican datos a un URL. El servidor web se hace cargo de inyectar los mensajes al sistema tradicional de entrega de correo que hemos descrito. Por razones de seguridad, también se pueden usar los protocolos web estándar. Estos protocolos se encargan del cifrado de las páginas web y no les importa si el contenido de éstas es un mensaje de correo.

7.3 WORLD WIDE WEB

La web, popularmente conocida como World Wide Web, es un marco arquitectónico para acceder a cierto contenido vinculado distribuido en millones de máquinas por toda Internet. En 10 años pasó de ser una manera de coordinar el diseño de los experimentos de física de alta energía en Suiza a la aplicación que millones de personas piensan que es “Internet”. Su enorme popularidad se deriva del hecho de que es fácil

de usar para los principiantes, además de que provee acceso mediante una interfaz gráfica a un enorme cúmulo de información sobre casi cualquier tema concebible, desde aborígenes hasta zoología.

La web (también conocida como WWW) comenzó en 1989 en el CERN, el Centro Europeo de Investigación Nuclear. La idea inicial era ayudar a los equipos grandes, a menudo con miembros en media docena de países y zonas horarias o más, a que colaboraran mediante el uso de un conjunto siempre cambiante de informes, planos, dibujos, fotografías y demás documentos producidos por los experimentos de partículas en física. La propuesta de una red de documentos vinculados surgió de Tim Berners-Lee, físico del CERN. El primer prototipo (basado en texto) estaba en operación 18 meses después. Una demostración pública en la conferencia Hypertext '91 atrapó la atención de otros investigadores, lo que llevó a Marc Andressen, de la Universidad de Illinois, a desarrollar el primer navegador gráfico. Se llamó Mosaic y se liberó en febrero de 1993.

El resto, como dicen, es historia. Mosaic fue tan popular que un año más tarde, Andreessen formó su propia compañía llamada Netscape Communications Corp., cuya meta era desarrollar software web. Durante los siguientes tres años, Netscape Navigator y Microsoft Internet Explorer sostuvieron una "guerra de navegadores", en donde cada uno trataba de capturar una mayor parte del nuevo mercado al agregar frenéticamente más características (y, por ende, más errores) que el otro.

Durante las décadas de 1990 y 2000, los sitios web y las páginas web, como se le dice al contenido web, crecieron en forma exponencial hasta que hubo millones de sitios y miles de millones de páginas. Una pequeña cantidad de estos sitios se hicieron muy populares. Esos sitios y las empresas que los respaldan son los que definen en gran parte la web y la forma en que las personas la experimentan en la actualidad. Algunos ejemplos son: una librería (Amazon, empezó en 1994, capitalización de mercado de $50 mil millones), un sitio de subastas en línea (eBay, 1995, $30 mil millones), un sitio de búsqueda (Google, 1998, $150 mil millones) y redes sociales (Facebook, 2004, empresa privada valuada en más de $15 mil millones). Incluso el periodo que atraviesa el año 2000, cuando el valor de muchas empresas web aumentó hasta cientos o millones de dólares de la noche a la mañana, sólo para derrumbarse por completo al siguiente día cuando resultaron ser un mero despliegue publicitario, tiene su nombre. Se conoce como la **Era punto com**. Aún siguen surgiendo nuevas ideas que se convierten en un negocio multimillonario en la web. Muchas de ellas provienen de estudiantes. Por ejemplo, Mark Zuckerberg era un estudiante de Harvard cuando empezó Facebook, Sergey Brin y Larry Page eran estudiantes en Stanford cuando empezaron Google. Tal vez usted descubrirá la siguiente idea multimillonaria.

En 1994, el CERN y el MIT firmaron un acuerdo para establecer el **W3C** (**Consorcio World Wide Web**, del inglés *World Wide Web Consortium*), una organización dedicada al desarrollo de web, a la estandarización de protocolos y a fomentar la interoperabilidad entre los sitios. Berners-Lee se convirtió en el director. Desde entonces, cientos de universidades y compañías se han unido al consorcio. Aunque ahora hay más libros sobre la web de los que podemos contar, el mejor lugar para recibir información actualizada sobre ella es (naturalmente) la web misma. La página de inicio del consorcio se encuentra en *www.w3.org*. Los lectores interesados pueden encontrar ahí vínculos con páginas que cubren todos los numerosos documentos y actividades del consorcio.

7.3.1 Panorama de la arquitectura

Desde el punto de vista del usuario, la web consiste en una enorme colección de contenido en forma de **páginas web**, por lo general, conocidas simplemente como **páginas**. Cada una puede contener vínculos a otras páginas en cualquier lugar del mundo. Para seguir un vínculo, los usuarios pueden hacer clic en él, y a continuación los llevará a la página apuntada. Este proceso se puede repetir de manera indefinida. La idea de hacer que una página apunte a otra, lo que ahora se conoce como **hipertexto**, fue inventada por un profesor visionario de ingeniería eléctrica del MIT, Vannevar Bush, en 1945 (Bush, 1945). Esto fue

mucho antes de que se inventara Internet. De hecho, fue antes de que existieran las computadoras comerciales, aunque varias universidades habían producido prototipos que ocupaban habitaciones extensas pero tenían menos poder que una moderna calculadora de bolsillo.

Por lo general, las páginas se ven mediante un programa llamado **navegador**. Firefox, Internet Explorer y Chrome son ejemplos de navegadores populares. El navegador obtiene la página solicitada, interpreta el contenido y despliega la página en pantalla con el formato adecuado. El contenido en sí puede ser una mezcla de texto, imágenes y comandos de formato, ya sea en forma de un documento tradicional u otras formas de contenido, como un video o programas que produzcan una interfaz gráfica con la que puedan interactuar los usuarios.

En la parte superior izquierda de la figura 7-18 se muestra la imagen de una página. Ésta es la página del Departamento de Ciencias Computacionales e Ingeniería, de la Universidad de Washington. Esta página muestra texto y elementos gráficos (que en su mayoría son muy pequeños como para leerlos). Algunas partes de la página están asociadas con vínculos a otras. A una pieza de texto, icono, imagen u otro elemento asociado con otra página se le conoce como hipervínculo. Para seguir un vínculo, el usuario coloca el cursor del ratón en la parte vinculada del área de la página (lo cual hace que el cursor cambie de forma) y hace clic. Seguir un vínculo es simplemente una forma de decir al navegador que obtenga otra página. En los primeros días de la web, los vínculos se resaltaban con texto subrayado y coloreado para que pudieran sobresalir. Hoy en día, los creadores de las páginas web tienen formas de controlar la apariencia de las regiones vinculadas, por lo que un vínculo podría aparecer como un icono o cambiar su apariencia cuando el ratón pase sobre él. Queda a criterio de los creadores de la página hacer que los vínculos se puedan distinguir en forma visual para proveer una interfaz que se pueda usar.

Los estudiantes en el Departamento pueden aprender más si siguen un vínculo a una página con información especial para ellos. Para acceder a este vínculo deben hacer clic en el área encerrada en un círculo. A continuación el navegador obtiene la nueva página y la despliega en pantalla, como se muestra de manera parcial en la parte inferior izquierda de la figura 7-18. Hay docenas de páginas adicionales a las que se puede acceder mediante vínculos desde la primera página, además de este ejemplo. Cualquier otra página

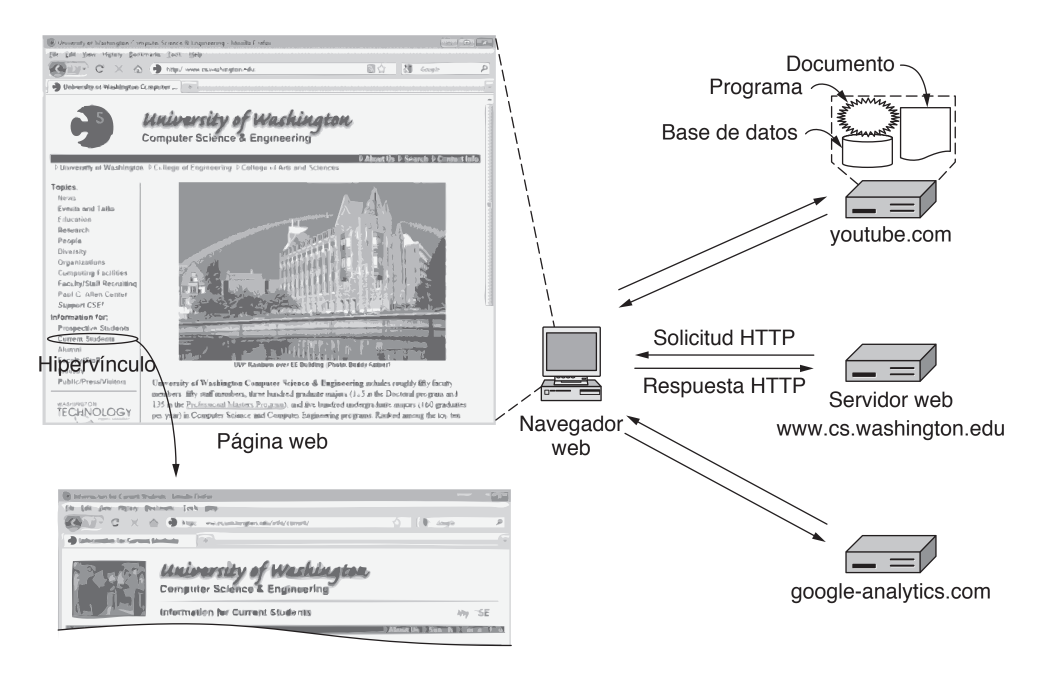


Figura 7-18. Arquitectura de la web.

puede estar compuesta de contenido en la(s) misma(s) máquina(s) que la primera página, o en máquinas en alguna parte del mundo. El usuario no puede distinguir esto. El navegador se encarga del proceso de obtención de las páginas, sin ninguna ayuda del usuario. De esta forma, el proceso de desplazarse entre máquinas al momento de ver contenido es transparente para el usuario.

En la figura 7-18 se muestra el modelo básico de la forma en que se despliegan las páginas en pantalla. El navegador despliega una página web en la máquina cliente. Para obtener cada página, se envía una solicitud a uno o más servidores, los cuales responden con el contenido de la página. El protocolo de solicitud-respuesta para obtener páginas es un protocolo simple basado en texto que se ejecuta sobre TCP, como en el caso de SMTP. Este protocolo se llama **HTTP** (**Protocolo de Transferencia de HiperTexto**, del inglés *HyperText Transfer Protocol*). El contenido puede ser simplemente un documento que se lea de un disco, o el resultado de una consulta en una base de datos y la ejecución de un programa. La página se considera una **página estática** si es el mismo documento cada vez que se despliega en pantalla. Por el contrario, si se generó bajo demanda mediante un programa o contiene uno, es una **página dinámica**.

Una página dinámica se puede presentar de manera distinta cada vez que se despliega en pantalla. Por ejemplo, la página principal de una tienda electrónica puede ser distinta para cada visitante. Si el cliente de una librería ha comprado novelas de misterio en sus anteriores visitas, es probable que la próxima vez le aparezcan las nuevas novelas de misterio en la página de inicio; mientras que un cliente enfocado hacia lo culinario podría ser recibido con nuevos libros de cocina. La manera en que el sitio web mantiene el registro de los gustos de cada cliente es algo que veremos en breve. En sí, la respuesta involucra el uso de *cookies*.

En la figura anterior, el navegador se contacta con tres servidores para obtener las dos páginas: *cs.washington.edu*, *youtube.com* y *google-analytics.com*. El contenido de estos distintos servidores se integra para que el navegador lo despliegue. La visualización conlleva un rango de procesamiento que depende del tipo de contenido. Además de generar texto y gráficos, puede implicar la reproducción de un video o la ejecución de una secuencia de comandos (*script*) que presente su propia interfaz de usuario como parte de la página. En este caso, el servidor *cs.washington.edu* suministra la página principal, el servidor *youtube.com* provee un video incrustado y el servidor *google-analytics.com* no suministra nada que el usuario pueda ver, pero rastrea los visitantes del sitio. Más adelante hablaremos con más detalle sobre los rastreadores.

El lado del cliente

Examinemos ahora el lado del navegador web en la figura 7-18 con más detalle. En esencia, un navegador es un programa que puede mostrar una página web y capturar clics del ratón para los elementos de la página visualizada. Al seleccionar un elemento, el navegador sigue el hipervínculo y obtiene la página seleccionada.

Cuando se creó la web por primera vez, resultó obvio a primera vista que para hacer que una página apuntara a otra página web se requerían mecanismos para nombrar y localizar páginas. En particular, había que responder a tres preguntas para desplegar una página:

1. ¿Cómo se llama la página?
2. ¿En dónde está ubicada?
3. ¿Cómo se puede acceder a ella?

Si a cada página se le asignara de alguna forma un nombre único, no habría ninguna ambigüedad a la hora de identificarlas. Sin embargo, esto no resolvería el problema. Considere un paralelismo entre las personas y las páginas. En Estados Unidos casi todos tienen un número de seguro social, el cual es un identificador único, ya que no debe haber dos personas con el mismo número. Pero si usted sólo cuenta

con el número de Seguro Social, no hay manera de averiguar la dirección del propietario ni tampoco puede saber si debe escribir a la persona en inglés, español o chino. La web tiene básicamente los mismos problemas.

La solución que se eligió identifica a las páginas de una manera que resuelve los tres problemas a la vez. A cada página se le asigna un **URL** (**Localizador Uniforme de Recursos**, del inglés *Uniform Resource Locator*) que sirva de manera efectiva como el nombre mundial de la página. Los URL tienen tres partes: el protocolo (también conocido como **esquema**), el nombre DNS de la máquina en la que se encuentra la página y la ruta que indica de manera única la página específica (un archivo a leer o un programa a ejecutar en la máquina). En el caso general, la ruta tiene un nombre jerárquico que modela la estructura de un directorio de archivos. Sin embargo, la interpretación de la ruta depende del servidor; puede o no reflejar la estructura del directorio real.

Como ejemplo, el URL de la página que se muestra en la figura 7-18 es:

http://www.cs.washington.edu/index.html

Este URL consiste de tres partes: el protocolo (*http*), el nombre DNS del host (*www.cs.washington.edu*) y el nombre de la ruta (*index.html*).

Cuando un usuario hace clic en un hipervínculo, el navegador lleva a cabo una serie de pasos para obtener la página a la que apunta. Vamos a rastrear los pasos que se realizan al momento de seleccionar nuestro vínculo de ejemplo:

1. El navegador determina el URL (al ver lo que se seleccionó).
2. El navegador pide al DNS la dirección IP del servidor *www.cs.washington.edu*.
3. El DNS responde con 128.208.3.88.
4. El navegador realiza una conexión TCP a 128.208.3.88 en el puerto 80, el puerto conocido para el protocolo HTTP.
5. Después envía una solicitud HTTP para pedir la página */index.html*.
6. El servidor *www.cs.washington.edu* envía la página como una respuesta HTTP, por ejemplo, enviando el archivo */index.html*.
7. Si la página incluye los localizadores URL necesarios para desplegar en pantalla, el navegador obtiene los otros URL mediante el mismo proceso. En este caso, los URL incluyen varias imágenes incrustadas que también se obtienen de *www.cs.washington.edu*, así como un video de *youtube.com* y una secuencia de comandos (*script*) de *google-analytics.com*.
8. El navegador despliega la página */index.html* como aparece en la figura 7-18.
9. Se liberan las conexiones TCP si no hay más solicitudes para los mismos servidores durante un periodo corto.

Muchos navegadores despliegan en una línea de estado, que se encuentra en la parte inferior de la pantalla, qué paso están ejecutando en ese momento. De esta manera, cuando el desempeño es pobre, el usuario puede ver si se debe a que el DNS o el servidor no están respondiendo, o simplemente hay congestionamiento en la Red durante la transmisión de la página.

El diseño del URL es abierto en el sentido en que los navegadores pueden usar con facilidad varios protocolos para llegar a distintos tipos de recursos. De hecho, se han definido localizadores URL para otros protocolos. En la figura 7-19 se muestra una lista de las formas simplificadas de los URL más comunes.

Ahora analicemos la lista de forma breve. El protocolo *http* es el lenguaje nativo de la web, al que hablan los servidores web. Lo examinaremos con más detalle más adelante en esta sección.

El protocolo *ftp* se utiliza para acceder a los archivos mediante FTP, el protocolo de transferencia de archivos de Internet. FTP ya existía antes de la web; ha estado en uso por más de tres décadas. La web facilita el proceso para obtener archivos colocados en muchos servidores FTP en todo el mundo al proveer

Nombre	Se usa para	Ejemplo
http	Hipertexto (HTML).	http://www.ee.uwa-edu/~rob/
https	Hipertexto con seguridad.	https://www.bank.com/ accounts/
ftp	FTP.	ftp://ftp.cs.vu.nl/pub/minix/README
file	Archivo local.	file://usr/suzanne/prog.c
mailto	Enviar correo electrónico.	mailto:JohnUsuario@acm.org
rtsp	Medios de flujo continuo.	rtsp://youtube.com/montypython.mpg
sip	Llamadas multimedia.	sip:eva@adversario.com
about	Información del navegador.	about:plugins

Figura 7-19. Algunos esquemas comunes de URL.

una interfaz simple en la que se puede hacer clic, en vez de una interfaz de línea de comandos. Este acceso mejorado a la información es una de las razones del crecimiento espectacular de la web.

Es posible acceder a un archivo local como una página web mediante el uso del protocolo *file*, o dicho en forma más simple, con sólo nombrarlo. Para usar este método no hay que tener un servidor. Desde luego que sólo funciona para los archivos locales, no para los remotos.

El protocolo *mailto* en realidad no tiene la habilidad de obtener páginas web, pero de todas formas es útil, ya que permite a los usuarios enviar correo electrónico desde un navegador web. La mayoría de los navegadores responderán cuando el usuario siga un vínculo *mailto* e iniciarán el agente de correo de ese usuario para redactar un mensaje con el campo de dirección llenado de antemano.

Los protocolos *rtsp* y *sip* son para establecer sesiones de medios de flujo continuo, y llamadas de audio y video.

Por último, el protocolo *about* es una convención que provee información sobre el navegador. Por ejemplo, si el usuario sigue el vínculo *about:plugins*, la mayoría de los navegadores mostrarán una página que lista los tipos MIME que pueden manejar, con las extensiones del navegador conocidas como complementos o *plug-ins*.

En resumen, los URL se diseñaron no sólo para permitir a los usuarios navegar en la web, sino también para ejecutar protocolos antiguos como FTP y el correo electrónico, así como los protocolos más recientes para audio y video, y para proveer un acceso conveniente a los archivos locales y la información del navegador. Esta metodología hace que todos los programas de interfaz de usuario especializados para esos otros servicios sean innecesarios, e integra casi todo el acceso a Internet en un solo programa: el navegador web. Si no fuera por el hecho de que un físico inglés que trabajaba en un laboratorio de investigación en Suiza concibió esta idea, podría pasar fácilmente como un plan soñado por el Departamento de Publicidad de una empresa de software.

A pesar de todas estas agradables propiedades, el uso creciente de la web ha descubierto una debilidad inherente en el esquema de URL. Un URL apunta a un host específico, pero algunas veces es conveniente hacer referencia a una página sin necesidad de decir al mismo tiempo en dónde está. Por ejemplo, en las páginas con muchas referencias es conveniente tener varias copias en ubicaciones distantes y separadas, para reducir el tráfico de red. No hay manera de decir: "Quiero la página *xyz*, pero no me importa de dónde la obtengas".

Para resolver este tipo de problema, los URL se generalizaron en **URI** (**Identificadores Uniformes de Recursos**, del inglés *Uniform Resource Identifiers*). Algunos URI indican cómo localizar un recurso. Éstos son los URL. Otros URI indican el nombre de un recurso, pero no en dónde encontrarlo. Estos URI se co-

nocen como **URN** (**Nombres Uniformes de Recursos**, del inglés *Uniform Resource Names*). Las reglas para escribir nombres URI se proporcionan en el RFC 3986, mientras que la IANA rastrea los distintos esquemas URI en uso. Hay muchos tipos de URI, además de los esquemas que se listan en la figura 7-19, pero éstos son los que dominan la web según su uso actual.

Tipos MIME

Para desplegar la nueva (o cualquier) página, el navegador tiene que comprender su formato. Para que todos los navegadores puedan entender todas las páginas, éstas deben escribirse en un lenguaje común, conocido como HTML, el estándar de la web (por ahora). Analizaremos con detalle este lenguaje más adelante.

Aunque un navegador es en esencia un intérprete de HTML, la mayoría tiene muchos botones y características para facilitar el proceso de navegación; entre ellos un botón para regresar a la página anterior y otro para avanzar a la siguiente (que sólo se pueden accionar después de que el usuario haya abierto por lo menos un par de ellas), y otro para ir a la página de inicio establecida por el usuario. La mayoría de los navegadores tienen un botón o elemento de menú para establecer un favorito en una página dada, y otro para mostrar la lista de favoritos, de modo que sea posible volver a visitar esos sitios con sólo unos cuantos clics del ratón.

Como indica nuestro ejemplo, las páginas HTML pueden contener elementos de contenido complejos y no sólo texto e hipertexto. Para una generalidad adicional, no todas las páginas necesitan contener HTML. Una página puede consistir de un video en formato MPEG, un documento en formato PDF, una fotografía en formato JPEG, una canción en formato MP3 o cualquiera de cientos de otros tipos de archivos. Como las páginas HTML estándar pueden vincular a cualquiera de estos elementos, el navegador tiene un problema cuando llega a una página que no sabe cómo interpretar.

En vez de hacer los navegadores cada vez más grandes al construir intérpretes para una colección extensa de tipos de archivos que aumenta con rapidez, la mayoría de los navegadores han elegido una solución más general. Cuando un servidor devuelve una página, también devuelve cierta información adicional sobre ella. Esta información incluye el tipo MIME de la página (vea la figura 7-13). Las páginas de tipo *text/html* sólo se despliegan directamente, al igual que las páginas en unos cuantos tipos integrados más. Si el tipo MIME no es de los integrados, el navegador consulta su tabla de tipos MIME para determinar cómo desplegar la página. Esta tabla asocia los tipos MIME con los visores.

Existen dos posibilidades: complementos (*plug-ins*) y aplicaciones auxiliares. Un complemento es un módulo de código de terceros que se instala como una extensión para el navegador, como se muestra en la figura 7-20(a). Algunos ejemplos comunes son los complementos para PDF, Flash y Quicktime, de modo que se puedan desplegar los documentos, además de reproducir audio y video. Como los complementos se ejecutan dentro del navegador, tienen acceso a la página actual y pueden modificar su apariencia.

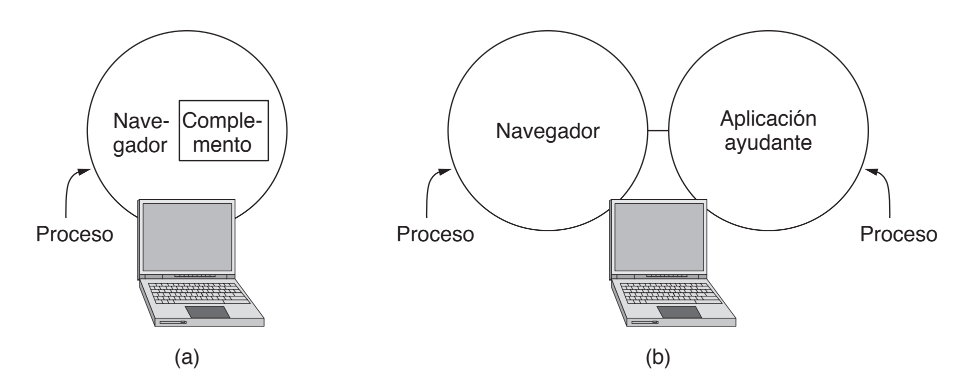


Figura 7-20. (a) Complemento del navegador. (b) Una aplicación ayudante.

Cada navegador tiene un conjunto de procedimientos que todos los complementos deben implementar para que el navegador pueda llamarlos. Por ejemplo, por lo general hay un procedimiento que el código base del navegador invoca para proveer al complemento los datos a visualizar. Este conjunto de procedimientos es la interfaz del complemento; es específica para cada navegador.

Además, el navegador crea un conjunto de sus propios procedimientos disponibles para el complemento, para que pueda proveer servicios a otros complementos. Los procedimientos comunes en la interfaz del navegador son para asignar y liberar memoria, desplegar un mensaje en la línea de estado del navegador y consultar a éste sobre los parámetros.

Antes de poder usar un complemento, hay que instalarlo. El procedimiento usual de instalación es que el usuario vaya al sitio web del complemento y descargue un archivo de instalación. Al ejecutar este archivo se desempaca el complemento y se hacen las llamadas apropiadas para registrar el tipo MIME del complemento con el navegador, además de asociarlo con el complemento. Por lo general los navegadores vienen precargados con los complementos populares.

La otra forma de extender un navegador es utilizar una **aplicación ayudante**, la cual es un programa completo que se ejecuta como un proceso separado y se ilustra en la figura 7-20(b). Como el ayudante es un programa separado, la interfaz se mantiene a distancia del navegador. Por lo general sólo acepta el nombre de un archivo de trabajo en el que se almacena el contenido, abre ese archivo y muestra el contenido en pantalla. Por lo general, los ayudantes son programas extensos que existen de manera independiente del navegador; por ejemplo, Microsoft Word o PowerPoint.

Muchas aplicaciones ayudantes usan el tipo MIME *application*. Como consecuencia se ha definido un número considerable de subtipos para que los utilicen; por ejemplo, *application/vnd.ms-powerpoint* para los archivos de PowerPoint. El término *vnd* indica formatos específicos de cada distribuidor. De esta manera, un URL puede apuntar directamente a un archivo de PowerPoint y, cuando el usuario haga clic en él, se iniciará ese programa de manera automática y se le entregará el contenido a desplegar. Las aplicaciones ayudantes no están restringidas al uso del tipo MIME *application*. Por ejemplo, Adobe Photoshop usa *image/x-photoshop*.

Asimismo, los navegadores se pueden configurar para manejar una cantidad prácticamente ilimitada de tipos de documentos sin necesidad de hacer cambios en ellos. Los servidores web modernos se configuran por lo general con cientos de combinaciones de tipo/subtipo y se agregan nuevas cada vez que se instala un nuevo programa.

Una posible fuente de conflictos es el hecho de que hay varios complementos y aplicaciones ayudantes disponibles para algunos subtipos, como *video/mpeg*. Lo que ocurre es que el último en registrarse sobrescribe la asociación existente con el tipo MIME y captura el tipo para sí mismo. Esto quiere decir que si se instala un nuevo programa, tal vez cambie la forma en que un navegador maneja los tipos existentes.

Los navegadores también pueden abrir archivos locales, si no hay una red a la vista, en vez de obtenerlos de servidores web remotos. Sin embargo, el navegador necesita alguna forma de determinar el tipo MIME del archivo. El método estándar es que el sistema operativo asocie una extensión de archivo con un tipo MIME. En una configuración típica, al abrir *foo.pdf* se abrirá en el navegador mediante el uso de un complemento *application/pdf* y al abrir *bar.doc* se abrirá en Word como la aplicación ayudante *application/msword*.

Aquí también pueden surgir conflictos, ya que muchos programas están dispuestos a (mejor dicho, deseosos de) manejar, por ejemplo, los archivos mpg. Durante la instalación, los programas destinados para usuarios sofisticados muestran con frecuencia casillas de verificación para los tipos MIME y las extensiones que están preparados a manejar, para permitir al usuario que seleccione los apropiados y de esta forma no sobrescriba las asociaciones existentes por accidente. Los programas orientados al mercado del consumidor suponen que el usuario no tiene idea de lo que es un tipo MIME, por lo que simplemente agarran todo lo que pueden sin importar lo que hayan hecho los programas que se instalaron antes.

La habilidad de extender el navegador con una gran cantidad de tipos nuevos es conveniente, pero también puede provocar problemas. Cuando un navegador en una PC con Windows obtiene un archivo con la extensión *exe*, se da cuenta de que este archivo es un programa ejecutable y, por lo tanto, no tiene ayudante. La acción obvia es ejecutar el programa. Sin embargo, esto podría ser un enorme hueco de seguridad. Todo lo que tiene que hacer un sitio web malicioso es producir una página web con imágenes de, por decir, estrellas de cine o astros del deporte, todas ellas con vínculos a un virus. Así, un solo clic en una de las imágenes provoca que se descargue y ejecute un programa ejecutable desconocido y potencialmente hostil en la máquina del usuario. Para prevenir invitados no deseados como éste, Firefox y otros navegadores vienen configurados para tener cuidado al ejecutar programas desconocidos de manera automática, pero no todos los usuarios comprenden qué opciones son seguras en vez de convenientes.

El lado del servidor

Ya vimos los detalles sobre el lado del cliente. Ahora demos un vistazo al lado del servidor. Como vimos antes, cuando el usuario escribe un URL o hace clic en una línea de hipertexto, el navegador analiza el URL e interpreta la parte entre *http://* y la siguiente barra diagonal como un nombre DNS que debe buscar. Equipado con la dirección IP del servidor, el navegador establece una conexión TCP con el puerto 80 de ese servidor. Después envía un comando que contiene el resto del URL, que es la ruta a la página en ese servidor. A continuación, el servidor devuelve la página para que el navegador la despliegue en pantalla.

A primera instancia, un servidor web simple es similar al servidor de la figura 6-6; el cual recibe el nombre de un archivo que debe buscar y devolver a través de la red. En ambos casos, los pasos que el servidor realiza en su ciclo principal son:

1. Aceptar una conexión TCP de un cliente (un navegador).
2. Obtener la ruta a la página, que viene siendo el nombre del archivo solicitado.
3. Obtener el archivo (del disco).
4. Enviar el contenido del archivo al cliente.
5. Liberar la conexión TCP.

Los servidores web modernos tienen más características, pero en esencia esto es lo que hace un servidor web para el caso simple del contenido que se encuentra en un archivo. En cuanto al contenido dinámico, el tercer paso se puede reemplazar por la ejecución de un programa (que se determina con base en la ruta) que devuelve el contenido.

Sin embargo, los servidores web se implementan con un diseño diferente para atender muchas solicitudes por segundo. Un problema con el diseño simple es que el proceso para acceder a los archivos es comúnmente el “cuello de botella”. Las lecturas de disco son muy lentas en comparación con la ejecución de un programa; además los mismos archivos se pueden leer repetidas veces del disco mediante el uso de las llamadas del sistema operativo. Otro problema es que sólo se procesa una solicitud a la vez. El archivo puede ser extenso y bloqueará a las otras solicitudes mientras se esté transfiriendo.

Una mejora evidente (que utilizan todos los servidores web) es mantener una caché en memoria de los *n* archivos leídos más recientes, o un cierto número de gigabytes de contenido. Así, antes de ir al disco para obtener un archivo, el servidor revisa la caché. Si el archivo está ahí, puede servirse directamente de la memoria, con lo cual se elimina el acceso al disco. Aunque un uso efectivo de la caché requiere de una gran cantidad de memoria principal y cierto tiempo de procesamiento adicional para verificar la caché y administrar su contenido, el ahorro en tiempo justifica casi siempre la sobrecarga y los gastos implícitos.

Para lidiar con el problema de atender una sola solicitud a la vez, una estrategia es hacer al servidor **multihilos**. En un diseño, el servidor consiste en un módulo de *front-end* que acepta todas las solicitudes

entrantes y *k* módulos de procesamiento, como se muestra en la figura 7-21. Los *k* + 1 hilos pertenecen al mismo proceso, por lo que todos los módulos de procesamiento tienen acceso a la caché dentro del espacio de direcciones del proceso. Cuando llega una solicitud, el *front-end* la acepta y construye un registro corto para describirla. Después entrega el registro a uno de los módulos de procesamiento.

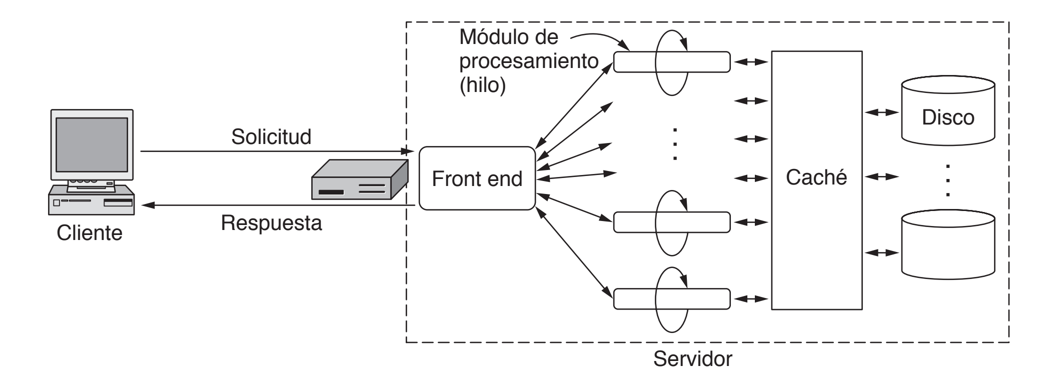


Figura 7-21. Un servidor web multihilos con un *front-end* y módulos de procesamiento.

El módulo de procesamiento primero verifica la caché para ver si el archivo solicitado está ahí. De ser así, actualiza el registro para incluir un apuntador al archivo en el registro. Si no está el módulo de procesamiento inicia una operación de disco para leerlo y colocarlo en la caché (es posible que descarte algunos otros archivos en caché para hacerle espacio). Cuando el archivo llega del disco, se coloca en la caché y también se regresa al cliente.

La ventaja de este esquema es que mientras uno o más módulos de procesamiento están bloqueados esperando a que termine una operación del disco o de red (y, por lo tanto, no consumen tiempo del CPU), otros módulos pueden estar trabajando activamente en otras solicitudes. Con *k* módulos de procesamiento, la tasa de transferencia puede ser hasta *k* veces más alta que con un servidor de un solo hilo. Por supuesto que, cuando el disco o la red son el factor limitante, es necesario tener múltiples discos o una red más veloz con el fin de obtener una mejora real en comparación con el modelo de un solo hilo.

Los modernos servidores web hacen más que sólo aceptar nombres de rutas y regresar archivos. De hecho, el procesamiento real de cada solicitud puede ser muy complicado. Por esta razón, en muchos servidores cada módulo de procesamiento realiza una serie de pasos. El *front-end* pasa cada solicitud entrante al primer módulo disponible, que a su vez la atiende mediante el uso de algún subconjunto de los siguientes pasos, dependiendo de los que sean necesarios para esa solicitud en particular. Estos pasos ocurren después de haber establecido la conexión TCP y algún mecanismo de transporte seguro (como SSL/TLS, que describiremos en el capítulo 8).

1. Resuelve el nombre de la página web solicitada.
2. Realiza control de acceso en la página web.
3. Verifica la caché.
4. Obtiene del disco la página solicitada o ejecuta un programa para construirla.
5. Determina el resto de la respuesta (por ejemplo, el tipo MIME a enviar).
6. Regresa la respuesta al cliente.
7. Realiza una entrada en el registro del servidor.

El paso 1 es necesario porque la solicitud entrante tal vez no contenga el nombre verdadero del archivo o programa como una cadena literal. Tal vez contenga accesos directos integrados que haya que traducir.

Como un ejemplo simple, el URL *http://www.cs.vu.nl/* tiene un nombre de archivo vacío. Hay que expandirlo a cierto nombre de archivo predeterminado, que por lo general es *index.html*. Otra regla común es asociar *~usuario/* al directorio web del *usuario*. Estas reglas se pueden usar juntas. Así, la página de inicio de uno de los autores (AST) se puede visitar en

http://www.cs.vu.nl/~ast/

aun cuando el verdadero nombre de archivo es *index.html* en cierto directorio predeterminado.

Además, los navegadores modernos pueden especificar información de configuración, como el software del navegador y el lenguaje predeterminado del usuario (por ejemplo, italiano o inglés). Esto hace posible que el servidor seleccione una página web con pequeñas imágenes para un dispositivo móvil y en el lenguaje preferido, si está disponible. En general, la expansión de nombres no es tan trivial como podría parecer a simple vista, debido a una variedad de convenciones sobre cómo asignar rutas al directorio de archivos y a los programas.

El paso 2 consiste en verificar si se cumple alguna restricción de acceso asociada con la página. No todas las páginas están disponibles para el público en general. Determinar cuándo un cliente puede obtener una página, puede depender de la identidad del cliente (por ejemplo, con base en los nombres de usuario y contraseñas) o de la ubicación del cliente en el DNS o espacio IP. Por ejemplo, una página puede estar restringida para los usuarios dentro de una empresa. La forma en que se logra esto depende del diseño del servidor. Es decir, para el popular servidor Apache la convención es colocar un archivo llamado *.htaccess* que lista las restricciones de acceso en el directorio en donde se encuentra la página restringida.

Los pasos 3 y 4 tratan sobre cómo obtener la página. El hecho de que se pueda obtener o no de la caché depende de las reglas de procesamiento. Por ejemplo, las páginas que se crean mediante la ejecución de programas no siempre se pueden poner en la caché, puesto que podrían producir un resultado distinto cada vez que se ejecuten. Incluso los archivos deberían verificarse de vez en cuando para ver si su contenido ha cambiado, de modo que se pueda eliminar el contenido antiguo de la caché. Si la página requiere que se ejecute un programa, existe también el problema de establecer los parámetros del programa o la entrada. Estos datos provienen de la ruta o de otras partes de la solicitud.

El paso 5 consiste en determinar otras partes de la respuesta que acompañan al contenido de la página. El tipo MIME es un ejemplo. Puede provenir de la extensión del archivo, de las primeras palabras archivo o la salida del programa, de un archivo de configuración y de otras posibles fuentes.

El paso 6 consiste en regresar la página a través de la red. Para incrementar el desempeño, un cliente y un servidor pueden usar una sola conexión TCP para obtener varias páginas. Esta reutilización significa que se requiere cierta lógica para asociar una solicitud a una conexión compartida y devolver cada respuesta de modo que esté asociada con la solicitud correcta.

El paso 7 realiza una entrada en el registro del sistema para fines administrativos, además de mantener cualquier otra estadística importante. Dichos registros se pueden usar después para extraer información valiosa sobre el comportamiento del usuario; por ejemplo, el orden en el que las personas acceden a las páginas.

Cookies

Para navegar en la web como lo hemos descrito hasta ahora, es necesario obtener una serie de páginas independientes. No existe el concepto de un inicio de sesión. El navegador envía una solicitud a un servidor y recibe un archivo. Después el servidor se olvida de haber siquiera visto a ese cliente específico.

Este modelo es perfectamente adecuado para recuperar documentos disponibles al público, y funcionaba bien durante los primeros días después de que se creó la web. Sin embargo, no es apto para regresar distintas páginas a distintos usuarios dependiendo de lo que ya hayan realizado con el servidor. Este comportamiento es necesario para muchas interacciones continuas con los sitios web. Por ejemplo, algunos sitios web

(como los periódicos) requieren que los clientes se registren (y tal vez que paguen una mensualidad) para usarlos. Aquí surge la pregunta sobre cómo pueden los servidores diferenciar las solicitudes de los usuarios previamente registrados de las solicitudes de todos los demás. Un segundo ejemplo proviene del comercio electrónico. Si un usuario navega por una tienda electrónica, enviando artículos a su carrito de compras virtual de vez en cuando, ¿cómo lleva el servidor la cuenta del contenido del carrito? Un tercer ejemplo es la personalización de portales web como Yahoo! Los usuarios pueden establecer una página inicial personalizada detallada con sólo la información que desean (por ejemplo, sus acciones y sus equipos deportivos favoritos), pero ¿cómo puede el servidor mostrar la página correcta si no sabe quién es el usuario?

En primera instancia podríamos pensar que los servidores serían capaces de rastrear a los usuarios si observaran sus direcciones IP. Sin embargo, esta idea no funciona. Muchos usuarios comparten computadoras, en especial en el hogar, y la dirección IP simplemente identifica a la computadora, no al usuario. Peor aún, muchas empresas usan NAT, de modo que los paquetes portan la misma dirección IP para todos los usuarios. Es decir, todas las computadoras detrás de la caja NAT se ven igual para el servidor. Y muchos ISP asignan direcciones IP a los clientes mediante DHCP. Las direcciones IP cambian con el tiempo, por lo que para un servidor, tal vez usted, podría de repente parecerse a su vecino. Por todas estas razones, el servidor no puede usar direcciones IP para rastrear a los usuarios.

Para resolver este problema se utiliza un mecanismo que se ha criticado muchas veces: las **cookies**. El nombre se deriva de la antigua jerga de los programadores, en la que un programa llama a un procedimiento y recibe algo que tal vez necesite presentar después para realizar cierto trabajo. En este sentido, podemos considerar a un descriptor de archivos de UNIX o a un manejador de objeto de Windows como una cookie. Las cookies se implementaron por primera vez en el navegador Netscape en 1994, y ahora se especifican en el RFC 2109.

Cuando un cliente solicita una página web, el servidor puede proveer información adicional en forma de una cookie, junto con la página solicitada. La cookie es una cadena con nombre muy pequeña (a lo más de 4 KB) que el servidor puede asociar con un navegador. Esta asociación no es lo mismo que un usuario, pero es mucho más parecida y útil que una dirección IP. Los navegadores almacenan las cookies ofrecidas durante un intervalo, por lo general en un directorio de cookies en el disco del cliente, de modo que persistan entre una invocación y otra del navegador, a menos que el usuario haya deshabilitado las cookies. Éstas son sólo cadenas, no programas ejecutables. En principio, una cookie podría contener un virus, pero ya que se tratan como datos, no hay una manera oficial de que el virus se pueda ejecutar y hacer daño. No obstante, siempre es posible que algún hacker explote un error del navegador para provocar la activación.

Una cookie puede tener hasta cinco campos, como se muestra en la figura 7-22. El *Dominio* indica de dónde viene la cookie. Se supone que los navegadores verifican que los servidores no mientan acerca de su dominio ya que cada uno puede almacenar hasta 20 cookies por cliente. La *Ruta* es una trayectoria en la estructura del directorio del servidor que identifica qué partes del árbol de archivos del servidor pueden utilizar la cookie. Por lo general es (/), lo que significa el árbol completo.

Dominio	Ruta	Contenido	Expira	Seguro
toms-casino.com	/	CustomerID=297793521	15-10-10 17:00	Sí
jills-store.com	/	Cart=1-00501;1-07031;2-13721	11-1-11 14:22	No
aportal.com	/	Prefs= Stk:CSCO+ORCL;Spt:Jets	31-12-20 23:59	No
sneaky.com	/	UserID=4627239101	31-12-19 23:59	No

Figura 7-22. Algunos ejemplos de cookies.

El campo *Contenido* toma la forma *nombre* = *valor*. Tanto *nombre* como *valor* pueden ser lo que el servidor desee. Este campo es donde se almacena el contenido de la cookie.

El campo *Expira* especifica cuándo caduca la cookie. Si este campo está ausente, el navegador la descarta al salir. Tal cookie se conoce como cookie **no persistente**. Si se proporciona una hora y una fecha, se dice que la cookie es **persistente** y se mantiene hasta que expira. Las horas y fechas de expiración se dan en el Tiempo del Meridiano de Greenwich. Para eliminar una cookie del disco duro de un cliente, un servidor simplemente la envía de nuevo, pero con una fecha pasada.

Por último, se puede establecer el campo *Seguro* para indicar que el navegador sólo puede regresar la cookie a un servidor mediante un transporte seguro, es decir, SSL/TLS (que describiremos en el capítulo 8). Esta característica se utiliza para comercio electrónico, actividades bancarias y otras aplicaciones seguras.

Ya vimos cómo se adquieren las cookies pero, ¿cómo se utilizan? Justo antes de que un navegador solicite una página a un sitio web, verifica su directorio de cookies para ver si el dominio al que está solicitando la página ya colocó alguna cookie. De ser así, todas las cookies colocadas por ese dominio, y sólo ese dominio, se incluyen en el mensaje de solicitud. Cuando el servidor las obtiene, puede interpretarlas de la forma que desee.

Examinemos algunos usos posibles para las cookies. En la figura 7-22, el sitio *toms-casino.com* estableció la primera cookie que se utiliza para identificar al cliente. Cuando éste regresa la siguiente semana para despilfarrar más dinero, el navegador envía la cookie de forma que el servidor sepa quién es. Una vez que el servidor cuenta con el ID del cliente, puede buscar el registro de éste en una base de datos y utilizar esta información para construir una página web apropiada para desplegar en pantalla. Dependiendo de los hábitos conocidos de apuestas del cliente, esta página podría consistir en una mano de póquer, un listado de las carreras de caballos del día o una máquina tragamonedas.

La segunda cookie proviene de *jills-store.com*. El escenario aquí es un cliente que está vagando por una tienda en busca de cosas buenas que comprar. Cuando dicho cliente encuentra una ganga y hace clic en ella, el servidor la agrega a su carrito de compras (que se mantiene en el servidor) y también crea una cookie que contiene el código del producto y envía la cookie de regreso al cliente. Conforme el cliente vaga por la tienda y hace clic en nuevas páginas, la cookie se regresa al servidor cada vez que se solicita una página nueva. Conforme se acumulan más compras, el servidor las agrega a la cookie. Por último, cuando el cliente hace clic en PASAR A LA CAJA, la cookie, que ahora contiene la lista completa de compras, se envía junto con la solicitud. De esta forma el servidor sabe con exactitud lo que el cliente desea comprar.

La tercera cookie es para un portal web. Cuando el cliente hace clic en un vínculo que lo lleva al portal, el navegador envía la cookie. Ésta le indica al portal que construya una página que contenga los precios de las acciones de Cisco y Oracle, así como los resultados de fútbol de los Jets de Nueva York. Puesto que una cookie puede tener un tamaño de hasta 4 KB, hay bastante espacio para incluir preferencias más detalladas, como encabezados de los periódicos, el clima local, ofertas especiales, etcétera.

Un uso más controversial de las cookies es el de rastrear el comportamiento en línea de los usuarios. Esto permite a los operadores de sitios web comprender cómo los usuarios navegan en sus sitios; además los anunciantes crean perfiles de los anuncios o sitios que ha visto un usuario específico. La controversia es que los usuarios por lo general no están conscientes de que se está rastreando su actividad, incluso con perfiles detallados y a través de sitios web sin relación aparente. Sin embargo, el **rastreo web** es un gran negocio. DoubleClick, que provee y rastrea anuncios, está clasificado entre los 100 sitios web más ocupados del mundo por la compañía de monitoreo Alexa. Google Analytics, que rastrea el uso de los sitios para los operadores, se utiliza en más de la mitad de los 100 000 sitios más ocupados en la web.

Es fácil para un servidor rastrear la actividad de los usuarios mediante las cookies. Suponga que un servidor desea mantener el registro de cuántos visitantes únicos ha tenido y cuántas páginas vio un visitante antes de salir del sitio. Cuando llega la primera solicitud no hay cookie que la acompañe, por lo que

el servidor regresa una cookie que contiene *Counter = 1*. Las visitas subsiguientes a las páginas en ese sitio regresarán la cookie al servidor. Cada vez se incrementa el contador y se regresa al cliente. Al llevar un registro de los contadores, el servidor puede ver cuántas personas salieron del sitio después de ver la primera página, cuántas vieron dos páginas y así sucesivamente.

Rastrear el comportamiento de navegación de los usuarios entre un sitio y otro es sólo un poco más complicado. Funciona de la siguiente manera. Una agencia de publicidad, digamos, Sneaky Ads, contacta sitios web importantes y coloca en sus páginas anuncios de los productos de sus clientes, por lo cual paga a los dueños de los sitios una cuota. En lugar de dar a los sitios un archivo GIF para que lo coloquen en cada página, les proporciona un URL para dicho propósito. Cada URL que dicha agencia maneja contiene un número único en la ruta, como

http://www.sneaky.com/382674902342.gif

Cuando un usuario visita por primera vez una página, *P*, que contiene un anuncio de éstos, el navegador obtiene el archivo HTML. A continuación el navegador inspecciona el archivo HTML y ve el vínculo al archivo de imagen en *www.sneaky.com*, por lo que envía ahí un mensaje en el que solicita la imagen. Se regresa un archivo GIF que contiene un anuncio, junto con una cookie que contiene un ID de usuario único, 4627239101 en la figura 7-22. Sneaky registra el hecho de que el usuario con este ID visitó la página *P*. Esto es fácil de hacer puesto que se hace referencia a la ruta solicitada (*382674902342.gif*) sólo en la página *P*. Por supuesto, el anuncio real puede aparecer en miles de páginas, pero cada vez con un nombre diferente. Sneaky probablemente cobre un par de centavos al fabricante del producto cada vez que envíe el anuncio.

Más tarde, cuando el usuario visita otra página web que contenga cualquiera de los anuncios de Sneaky, el navegador primero obtiene el archivo HTML del servidor. Después ve el vínculo a, digamos, *http://www.sneaky.com/193654919923.gif* y solicita ese archivo. Puesto que ya tiene una cookie del dominio *sneaky.com*, el navegador incluye una cookie de Sneaky que contiene el ID del usuario. Sneaky ahora conoce una segunda página que el usuario ha visitado.

A su debido tiempo, Sneaky puede construir un perfil detallado de los hábitos de navegación del usuario, aunque éste nunca haya hecho clic en ninguno de los anuncios. Por supuesto, aún no tiene el nombre del usuario (aunque tiene su dirección IP, lo cual debe ser suficiente para deducir el nombre a partir de otras bases de datos). Pero si el usuario alguna vez proporciona su nombre a algún sitio que coopere con Sneaky, estará disponible un perfil completo junto con un nombre para venderlo a quien desee comprarlo. La venta de esta información puede ser lo bastante rentable como para que Sneaky coloque más anuncios en más sitios web y, por lo tanto, para que recolecte más información.

Y si Sneaky desea ser "supersneaky", el anuncio no necesita ser el clásico anuncio de pancarta (*banner*). Un "anuncio" que conste de un solo píxel en el color de fondo (lo que lo hace invisible) tiene exactamente el mismo efecto que un anuncio de pancarta: requiere que el navegador obtenga la imagen GIF de 1×1 píxeles y le envíe todas las cookies que se originan en el dominio del píxel.

Las cookies se han convertido en un punto central de debate en relación con la privacidad en línea, debido al comportamiento de rastreo antes mostrado. La parte más insidiosa de todo es que muchos usuarios están totalmente inconscientes de esta recolección de información, e incluso pueden pensar que están seguros debido a que no hacen clic en ninguno de los anuncios. Por esta razón, muchos consideran las cookies que rastrean a los usuarios de un sitio a otro como **spyware**. Dé un vistazo a las cookies que su navegador ya tiene almacenadas. La mayoría de los navegadores desplegarán esta información junto con las preferencias de privacidad actuales. Tal vez le sorprenda encontrar nombres, direcciones de correo electrónico o contraseñas, así como identificadores opacos. Con suerte y no encontrará números de tarjetas de crédito, pero el potencial de abuso está claro.

Para mantener algo de su privacidad, algunos usuarios configuran sus navegadores para que rechacen todas las cookies. Sin embargo, esto puede causar problemas debido a que muchos sitios web no funcio-

nan bien sin cookies. Como alternativa, la mayoría de los navegadores permiten a los usuarios bloquear las **cookies de terceros**. Una cookie de tercero proviene de un sitio distinto al de la página principal que se obtiene; por ejemplo, la cookie de *sneaky.com* que se utiliza al interactuar con la página *P* en un sitio web completamente distinto. Bloquear estas cookies ayuda a evitar el rastreo entre un sitio web y otro. También se pueden instalar extensiones del navegador para proveer un control más detallado sobre la forma en que se usan las cookies (o más bien, que no se usan). A medida que continúa el debate, muchas empresas están desarrollando políticas privadas que limitan la forma en que compartirán información para evitar el abuso. Claro está que las políticas consisten simplemente en la forma como las empresas dicen que van a manejar la información. Por ejemplo: "Podemos usar la información que recolectamos sobre usted para llevar a cabo nuestro negocio", cuyo negocio podría ser vender la información.

7.3.2 Páginas web estáticas

La base de la web es transferir páginas web del servidor al cliente. En su forma más simple, las páginas web son estáticas. Es decir, son sólo archivos guardados en un servidor que se presentan a sí mismos de la misma forma cada vez que un cliente los obtiene y los ve. Sin embargo, el hecho de que son estáticas no significa que las páginas sean inertes en el navegador. Una página que contenga un video puede ser una página web estática.

Como dijimos antes, la lengua franca de la web, en la que se escriben la mayoría de las páginas, es HTML. Las páginas de inicio de los maestros por lo general son páginas HTML estáticas. Las páginas de inicio de las empresas son comúnmente páginas dinámicas creadas por una empresa de diseño web. En esta sección daremos un breve vistazo a las páginas HTML estáticas como base para el material posterior. Los lectores que ya estén familiarizados con el HTML pueden saltar hasta la siguiente sección, en donde describiremos el contenido dinámico y los servicios web.

HTML: Lenguaje de Marcado de HiperTexto

El **HTML** (**Lenguaje de Marcado de HiperTexto**, del inglés *HyperText Markup Language*) se introdujo con la web. Permite a los usuarios producir páginas web que incluyen texto, gráficos, video, apuntadores a otras páginas web y más. HTML es un lenguaje de marcado que sirve para describir cómo se va a dar formato a los documentos. El término "marcado" proviene de la época en que los correctores de estilo realmente marcaban los documentos para indicar a la imprenta —en aquellos tiempos, un humano— qué fuentes utilizar, y cosas por el estilo. Por lo tanto, los lenguajes de marcado contienen comandos explícitos para dar formato. Por ejemplo, en HTML, **significa iniciar modo en negritas** y significa abandonar modo en negritas. LaTeX y TeX son otros ejemplos de lenguajes de marcado bien conocidos para la mayoría de los autores académicos.

La ventaja clave de un lenguaje de marcado sobre uno con marcado no explícito es que separa el contenido de la forma en que se debe presentar. Así, escribir un navegador es algo directo: el navegador simplemente tiene que entender los comandos de marcado y aplicarlos al contenido. Al integrar todos los comandos de marcado dentro de cada archivo HTML y estandarizarlos, se hace posible que cualquier navegador web lea y vuelva a dar formato a cualquier página web. Esto es crucial porque una página se pudo haber producido en una ventana de 1600×1200 con colores de 24 bits en una computadora de gama alta, pero tal vez se vaya a desplegar en una ventana de 640×320 en un teléfono móvil.

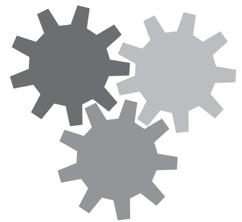
Aunque es posible escribir documentos como éstos con cualquier editor estándar de texto (y mucha gente lo hace), también es posible usar procesadores de texto o editores de HTML especiales que pueden hacer la mayoría del trabajo (pero que por lo mismo dan al usuario menos control sobre todos los detalles del resultado final).

En la figura 7-23 se muestra una página web simple escrita en HTML, y su presentación en un navegador. Una página web está constituida por un encabezado y un cuerpo, cada uno de los cuales está encerrado entre **etiquetas** (comandos de formato) <html> y </html>, aunque la mayoría de los navegadores no se quejan si faltan estas etiquetas. Como se puede ver en la figura 7-23(a), el encabezado está delimitado por las etiquetas <head> y </head>; y el cuerpo por <body> y </body>. Las cadenas dentro de

```
<html>
<head> <title> WIDGETS AMALGAMADOS, S.A. </title> </head>
<body> <h1> Bienvenidos a la página de inicio de WASA </h1>
<img src=”http://www.widget.com/imagenes/logo.gif” ALT=”Logo WASA”> <br>
Estamos muy contentos de que haya elegido visitar la página de inicio de <b>
Widgets Amalgamados </b>. Esperamos que <i> usted </i> encuentre aquí toda la información que necesita.
<p> A continuación presentamos vínculos con la información sobre nuestro surtido de productos finos. Puede
ordenar en forma electrónica (por WWW), por teléfono o por correo electrónico. </p>
<hr>
<h2> Información sobre nuestros productos </h2>
<ul>
   <li> <a href=”http://widget.com/productos/grandes”> Widgets grandes </a> </li>
   <li> <a href=”http://widget.com/productos/chicos”> Widgets chicos </a> </li>
</ul>
<h2> Información de contacto </h2>
<ul>
   <li> Teléfono: 1-800-WIDGETS </li>
   <li> Correo electrónico: info@widgets-amalgamados.com </li>
</ul>
</body>
</html>
```

(a)

Bienvenidos a la página de inicio de WASA



Estamos muy contentos de que haya elegido visitar la página de inicio de **Widgets Amalgamados**. Esperamos que usted encuentre aquí toda la información que necesita.

A continuación presentamos vínculos con la información sobre nuestro surtido de productos finos. Puede ordenar en forma electrónica (por WWW), por teléfono o por correo electrónico.

Información sobre nuestros productos

- Widgets grandes
- Widgets chicos

Información de contacto

- Por teléfono: 1-800-WIDGETS
- Correo electrónico: info@widget-amalgamados.com

Figura 7-23. (a) El HTML para una página web de ejemplo. (b) La página con formato.

las etiquetas se llaman **directivas**. La mayoría de las etiquetas HTML tiene este formato. Es decir, usan <algo> para marcar el comienzo de algo, y </algo> para marcar su fin.

Las etiquetas se pueden escribir tanto en minúsculas como en mayúsculas. Por lo tanto, <head> y <HEAD> significan lo mismo, pero es mejor usar minúsculas por cuestión de compatibilidad. La distribución real del documento HTML no es importante. Los analizadores de HTML ignoran los espacios extra y los retornos, puesto que tienen que volver a dar formato al texto para acomodarlo en el área de visualización actual. En consecuencia, se puede agregar espacio en blanco a voluntad para hacer más legibles los documentos HTML, algo que la mayoría necesita con urgencia. Otro resultado es la imposibilidad de usar líneas en blanco para separar párrafos, puesto que simplemente se ignoran. Se requiere una etiqueta explícita.

Algunas etiquetas tienen parámetros (con nombre), llamados **atributos**. Por ejemplo, la etiqueta <img> de la figura 7-23 se utiliza para incluir una imagen en línea con el texto. Tiene dos atributos: src y alt. El primero proporciona el URL para la imagen. El estándar HTML no especifica qué formatos de imagen se permiten. En la práctica, todos los navegadores soportan archivos GIF y JPEG. Los navegadores son libres de soportar otros formatos, pero esta extensión es una espada de doble filo. Si un usuario está acostumbrado a un navegador que soporta, por decir, archivos TIFF, puede incluirlos en sus páginas web para después sorprenderse cuando otros navegadores simplemente ignoren todo su maravilloso arte.

El segundo atributo proporciona el texto alternativo a usar si no se puede mostrar la imagen. Para cada etiqueta, el estándar de HTML proporciona una lista de cuáles son los parámetros permitidos (si los hay) y lo que significan. Como cada parámetro tiene un nombre, el orden en el que se dan no es importante.

Técnicamente, los documentos HTML se escriben en el conjunto de caracteres Latin-1 del ISO 8859-1, pero para los usuarios cuyos teclados sólo reconocen ASCII, hay secuencias de escape para los caracteres especiales, como è. La lista de caracteres especiales se proporciona en el estándar. Todos comienzan con un signo & y terminan con un punto y coma. Por ejemplo,   produce un espacio, è produce è y é produce é. Puesto que <, >, y & tienen significados especiales, se pueden expresar sólo con sus secuencias de escape, <, > y &, respectivamente.

El elemento principal del encabezado es el título, delimitado por <title> y </title>. También pueden estar presentes ciertos tipos de metainformación, aunque ninguno está presente en nuestro ejemplo. El título mismo no se despliega en la página. Algunos navegadores lo usan para rotular la ventana de la página.

En la figura 7-23 se utilizan varios encabezados. Cada encabezado se genera mediante una etiqueta <h*n*>, donde *n* es un dígito en el intervalo de 1 a 6. Por lo tanto, <h1> es el encabezado más importante y <h6> es el menos importante. Es responsabilidad del navegador desplegar estos encabezados de manera adecuada en la pantalla. Por lo general, los encabezados de menor número se presentarán con una fuente más grande y gruesa. El navegador también puede seleccionar colores distintos para cada nivel de encabezado. Por lo común, los encabezados <h1> son grandes y en negritas, con al menos una línea en blanco arriba y abajo. En contraste, los encabezados <h2> tienen una fuente más pequeña y con menos espacio arriba y abajo.

Las etiquetas <b> e <i> se usan para entrar en el modo de negritas y cursivas, respectivamente. La etiqueta <hr> inserta de manera obligatoria un salto de línea y dibuja una línea horizontal a través de la pantalla.

La etiqueta <p> inicia un párrafo. Para desplegarlo, el navegador podría insertar una línea en blanco y algo de sangría, por ejemplo. Como dato interesante, los programadores HTML perezosos omiten con frecuencia la etiqueta </p> que existe para marcar el final de un párrafo.

HTML proporciona varios mecanismos para crear listas, incluyendo las listas anidadas. Las listas desordenadas, como las que se muestran en la figura 7-23, inician con <ul> y <li> se usa para marcar el inicio de los elementos. También existe una etiqueta <ol> para iniciar una lista ordenada. Los elementos

individuales en las listas desordenadas aparecen por lo general con viñetas (•) al principio. El navegador se encarga de enumerar los elementos en las listas ordenadas.

Por último, llegamos a los hipervínculos. En la figura 7-23 podemos ver ejemplos de éstos, que utilizan las etiquetas <a> (ancla) y . La etiqueta <a> tiene varios parámetros, de los cuales el más importante es *href*, el URL vinculado. El texto entre <a> y se despliega en pantalla. Si se selecciona, se sigue el hipervínculo a una nueva página. También se permite vincular a otros elementos. Por ejemplo, se puede proporcionar una imagen entre <a> y mediante el uso de <img>. En este caso, la imagen se despliega y, al hacer clic en ella, se activa el hipervínculo.

Existen muchas otras etiquetas y atributos HTML que no hemos visto en este ejemplo simple. Por ejemplo, la etiqueta <a> puede tomar un parámetro *nombre* para plantar un hipervínculo, lo cual le permite apuntar a la parte media de una página. Esto es útil, por ejemplo, para páginas web que empiezan con una tabla de contenido en la que se puede hacer clic. Al hacer clic en un elemento en la tabla de contenido, el usuario salta a la sección correspondiente de la misma página. Un ejemplo de una etiqueta distinta es
, la cual obliga al navegador a interrumpir una línea y empezar una nueva.

Quizá la mejor manera de comprender las etiquetas sea verlas en acción. Para ello puede elegir una página web y analizar el HTML en su navegador, para ver cómo se elaboró esta página. La mayoría de los navegadores tienen un elemento de menú llamado VER CÓDIGO FUENTE (VIEW SOURCE), o algo similar. Al seleccionar este elemento se visualiza el código HTML de la página actual, en vez de su salida con formato.

Elaboramos un bosquejo de las etiquetas que han existido desde los principios de la web. El HTML sigue evolucionando. La figura 7-24 muestra partes de las características que se han agregado con versiones sucesivas de HTML. El HTML 1.0 se refiere a la versión de HTML utilizada al momento de introducir la web. Las versiones 2.0, 3.0 y 4.0 de HTML aparecieron en rápida sucesión en el transcurso de unos cuantos años, a medida que la web se expandió drásticamente. Después de HTML 4.0, pasó un periodo de casi 10 años para que se viera clara la ruta hacia la estandarización de la siguiente versión importante: HTML 5.0. Como es una actualización importante que consolida los medios por los que los navegadores manejan el contenido rico, el esfuerzo de HTML 5.0 es continuo y no se espera que produzca un estándar sino hasta después de 2012, a lo más. No obstante los estándares, los principales navegadores ya soportan la funcionalidad de HTML 5.0.

La progresión por las versiones de HTML trata sobre agregar nuevas características que la gente deseaba pero tenía que manejar en formas no estandarizadas (como los complementos) hasta que se hicieron estándar. Por ejemplo, HTML 1.0 y HTML 2.0 no tenían tablas. Se agregaron en el HTML 3.0. Una tabla de HTML consiste en una o más filas, cada una de las cuales consiste en una o más celdas en la tabla que contienen una amplia variedad de material (por ejemplo, texto, imágenes, otras tablas). Antes de HTML 3.0, los autores que necesitaban una tabla tenían que recurrir a métodos apropiados, como incluir una imagen que mostrara la tabla.

En el HTML 4.0 se agregaron más características nuevas. Éstas incluían características de accesibilidad para los usuarios discapacitados, la incrustación de objetos (una generalización de la etiqueta <img> de modo que también se puedan incrustar otros objetos en las páginas), soporte para los lenguajes de secuencias de comandos (para permitir contenido dinámico), y otras cosas más.

El HTML 5.0 incluye muchas características para manejar los medios ricos en contenido que ahora se utilizan de manera rutinaria en la web. Se puede incluir video y audio en las páginas para que lo reproduzca el navegador, sin que el usuario tenga que instalar complementos. Los dibujos se pueden acumular en el navegador como gráficos vectoriales, en vez de usar formatos de imágenes de mapas de bits (como JPEG y GIF). También hay más soporte para ejecutar secuencias de comandos en los navegadores, como los hilos de cálculos en segundo plano y acceso al almacenamiento. Todas estas características ayudan a soportar páginas web que sean más parecidas a las aplicaciones tradicionales con una interfaz de usuario que a los documentos. Ésta es la dirección que está siguiendo la web.

Elemento	HTML 1.0	HTML 2.0	HTML 3.0	HTML 4.0	HTML 5.0
Hipervínculos	x	x	x	x	x
Imágenes	x	x	x	x	x
Listas	x	x	x	x	x
Mapas e imágenes activas		x	x	x	x
Formularios		x	x	x	x
Ecuaciones			x	x	x
Barras de herramientas			x	x	x
Tablas			x	x	x
Características de accesibilidad				x	x
Incrustación de objetos				x	x
Hojas de estilo				x	x
Secuencias de comandos				x	x
Video y audio					x
Gráficos vectoriales en línea					x
Representación de XML					x
Hilos en segundo plano					x
Almacenamiento del navegador					x
Lienzo de dibujo					x

Figura 7-24. Algunas diferencias entre versiones de HTML.

Entrada y formularios

Hay una importante herramienta que aún no hemos examinado: la entrada de datos. En esencia, HTML 1.0 era de un solo sentido. Los usuarios podían obtener las páginas de información de los proveedores, pero era difícil enviar información en el otro sentido. Pronto se hizo obvio que se necesitaba tráfico de dos vías para colocar pedidos de productos a través de páginas web, llenar tarjetas de registro en línea, introducir términos de búsqueda y mucho, mucho más.

Para enviar entrada del usuario al servidor (a través del navegador) se requieren dos tipos de soporte. Primero, que HTTP pueda transportar datos en esta dirección. En una sección posterior describiremos cómo hacer esto; se utiliza el método *POST*. El segundo requerimiento es presentar elementos de la interfaz de usuario que recopilen y empaqueten la entrada. En el HTML 2.0 se incluyeron **formularios** con esta funcionalidad.

Los formularios contienen cuadros o botones que permiten a los usuarios proporcionar información o tomar decisiones, y después enviar dicha información al dueño de la página. Se escriben justo igual que otras partes de HTML, como podemos ver en el ejemplo de la figura 7-25. Debemos tener en cuenta que los formularios siguen siendo contenido estático. El contenido dinámico, que analizaremos más adelante, provee formas más sofisticadas de recopilar la entrada mediante el envío de un programa cuyo comportamiento puede depender del entorno del navegador.

```
<html>
<head> <title> FORMULARIO DE PEDIDO DE CLIENTES WASA </title> </head>
<body>
<h1> Orden de compra de widgets </h1>
<form ACTION=”http://widget.com/cgi-bin/pedido.cgi” method=POST>
<p> Nombre <input name=”cliente” size=46> </p>
<p> Dirección <input name=”direccion” size=40> </p>
<p> Ciudad <input name=”city” size=20> Estado <input name=”estado” size =4>
País <input name=”pais” size=10> </p>
<p> Núm. tarjeta de crédito <input name=”numtarj” size=10>
Expira <input name=”expira” size=4>
M/C <input name=”tc” type=radio value=”mastercard”>
VISA <input name=”tc” type=radio value=”visacard”> </p>
<p> Tamaño de widget Grande <input name=”producto” type=radio value=”costoso”>
Pequeño <input name=”producto” type=radio value=”barato”>
Enviar por mensajería rápida <input name=”express” type=checkbox> </p>
<p><input type=submit value=”Enviar pedido”> </p>
¡Gracias por ordenar un widget WASA, el mejor widget que el dinero puede comprar!
</form>
</body>
</html>
```

(a)

Orden de compra de widgets

Nombre

Dirección

Ciudad Estado País

Núm. tarjeta de crédito Expira M/C Visa

Tamaño de widget Grande Chico Enviar por mensajería rápida

Enviar pedido

¡Gracias por ordenar un widget WASA, el mejor widget que el dinero puede comprar!

(b)

Figura 7-25. (a) El HTML para un formulario de pedido. (b) La página con formato.

Al igual que todos los formularios, éste va encerrado entre las etiquetas <form> y </form>. Los atributos de esta etiqueta indican lo que se debe hacer con los datos que se introducen, en este caso mediante el método *POST* para enviar los datos al URL especificado. El texto que no va encerrado en una etiqueta simplemente se despliega en pantalla. Todas las etiquetas usuales (por ejemplo, <b>) se permiten en un formulario para que el autor de la página pueda controlar su apariencia en la pantalla.

En este formulario se utilizan tres tipos de cuadros de entrada, cada uno de los cuales usa la etiqueta <input>. Tiene una variedad de parámetros para determinar el tamaño, naturaleza y uso del cuadro

desplegado. Los formularios más comunes son campos en blanco para aceptar texto del usuario, cuadros que se pueden activar y botones *submit* que hacen que se devuelvan los datos al servidor.

El primer tipo de cuadro de entrada es un cuadro *text* que va después del texto "Nombre". El cuadro es de 46 caracteres de ancho y espera que el usuario escriba una cadena de texto, la cual a su vez se almacena en la variable llamada *cliente*.

En la siguiente línea del formulario se pide la dirección del usuario, de 40 caracteres de ancho. Después viene una línea que pide la ciudad, el estado y el país. Como no se utilizan etiquetas <p> entre estos campos, el navegador los muestra todos en una línea (en vez de mostrarlos como párrafos separados) si es que caben. En lo que respecta al navegador, el único párrafo contiene sólo seis elementos: tres cadenas de texto alternadas con tres cuadros. La siguiente línea pide el número de tarjeta de crédito y la fecha de expiración. Sólo hay que transmitir números de tarjetas de crédito a través de Internet cuando se hayan tomado las medidas de seguridad adecuadas. En el capítulo 8 hablaremos sobre esto.

Después de la fecha de expiración nos encontramos con una nueva herramienta: los botones de opción. Éstos se utilizan cuando hay que elegir entre dos o más alternativas. Aquí, el modelo intelectual es la radio de un automóvil con media docena de botones para elegir estaciones. Al hacer clic en un botón se desactivan todos los demás botones del mismo grupo. La presentación visual depende del navegador. El tamaño de *widget* también usa dos botones de opción. Los dos grupos se diferencian mediante su parámetro *nombre* y no mediante un alcance estático en el que se utiliza algo como <radiobutton> ... </radiobutton>.

Los parámetros *value* se utilizan para indicar qué botón de opción se oprimió. Por ejemplo, dependiendo de las opciones de tarjeta de crédito que haya elegido el usuario, se establecerá la variable *cc* con la cadena "mastercard" o la cadena "visacard".

Después de los dos conjuntos de botones de opción, llegamos a la opción de envío, representada por un cuadro de tipo *checkbox* (casilla de verificación). Puede estar activado o desactivado. A diferencia de los botones de opción, en donde debe estar seleccionado exactamente uno del conjunto, cada cuadro de tipo *checkbox* puede estar activado o desactivado en forma independiente de los otros.

Finalmente llegamos al botón enviar (*submit)*. La cadena *value* es la etiqueta del botón y se despliega en pantalla. Cuando el usuario hace clic en el botón *enviar*, el navegador empaqueta la información recolectada en una sola línea larga y la envía de vuelta al servidor, al URL que se proporciona como parte de la etiqueta <form>. Se utiliza una codificación simple. El signo & se usa para separar campos y el signo + se usa para representar el espacio. En nuestro formulario de ejemplo, la línea se podría ver como el contenido de la figura 7-26.

```
cliente=John+Doe&direccion=100+Main+St.&ciudad=White+Plains&
estado=NY&pais=USA&numtarj=1234567890&expira=6/14&cc=mastercard&
producto=barato&express=on
```

Figura 7-26. Una posible respuesta del navegador para el servidor, con la información escrita por el usuario.

La cadena se envía de vuelta al servidor como una línea (aquí se divide en tres líneas debido a que la página no tiene el ancho suficiente). Es responsabilidad del servidor dar sentido a esta cadena; lo más probable es que pase la información a un programa que la procese. En la siguiente sección veremos cómo se puede realizar esto.

Existen también otros tipos de entrada que no se muestran en este ejemplo simple. Dos de estos otros tipos son *password* y *textarea*. Un cuadro *password* es igual que un cuadro *text* (el tipo predeterminado que no necesita nombrarse), excepto que los caracteres no se muestran a medida que se escriben. Un cuadro *textarea* es también lo mismo que un cuadro *text*, sólo que puede contener varias líneas.

Para las listas largas en las que haya que elegir una opción, se proveen las etiquetas <select> y </select> para agrupar una lista de alternativas. A menudo esta lista se visualiza como un menú desplegable. La semántica es como la de los botones de opción, a menos que se proporcione el parámetro *multiple*, en cuyo caso la semántica es como la de las casillas de verificación.

Finalmente, hay maneras de indicar valores predeterminados o iniciales que el usuario puede modificar. Por ejemplo, si a un cuadro *text* se le asigna un campo *value*, el contenido se despliega en el formulario para que el usuario lo edite o borre.

CSS: Hojas de Estilo en Cascada

El objetivo original del HTML era especificar la *estructura* del documento, no su *apariencia*. Por ejemplo,

```
<h1>Fotos de Débora</h1>
```

indica al navegador que debe enfatizar el encabezado, pero no dice nada sobre el tipo de letra, tamaño de punto o color. Esto queda a discreción del navegador, que conoce las propiedades de la pantalla (por ejemplo, cuántos píxeles tiene). Sin embargo, muchos diseñadores de páginas web querían un control absoluto sobre la forma en que aparecían sus páginas, por lo que se agregaron nuevas etiquetas a HTML para controlar la apariencia, como:

```
<font face=”helvética” size=”24” color=”red”>Fotos de Débora</font>
```

Además se agregaron formas de controlar el posicionamiento en la pantalla con precisión. El problema con este método es que es tedioso y produce HTML inflado que no es portable. Aunque una página se puede visualizar perfectamente en el navegador en el que se desarrolló, puede ser un completo desastre en otro navegador, en otra versión del mismo navegador o a una resolución de pantalla distinta.

Una mejor alternativa es el uso de las hojas de estilo. En los editores de texto, las hojas de estilo permiten a los autores asociar texto con un estilo lógico en vez de uno físico; por ejemplo, "párrafo inicial" en vez de "texto en cursiva". La apariencia de cada estilo se define por separado. De esta manera, si el autor decide cambiar los párrafos iniciales de cursiva a 14 puntos en color azul, a negritas a 18 puntos en rosa impactante, todo lo que requiere es modificar una definición para convertir todo el documento.

El concepto de **CSS** (**Hojas de Estilo en Cascada**, del inglés *Cascading Style Sheets*) introdujo las hojas de estilo a la web con el HTML 4.0, aunque no se empezaron a popularizar y los navegadores no ofrecieron soporte sino hasta el año 2000. CSS define un lenguaje simple para describir reglas que controlan la apariencia de contenido etiquetado. Veamos un ejemplo. Suponga que WASA desea páginas web elegantes con texto color azul marino en fuente Arial sobre un fondo blanco, y encabezados de nivel que son 100% y 50% más grandes que el texto para cada nivel, respectivamente. La definición de CSS en la figura 7-27 proporciona estas reglas.

```
body {background-color:linen; color:navy; font-family:Arial;}
h1 {font-size:200%}
h2 {font-size:150%}
```

Figura 7-27. Ejemplo de CSS.

Como vemos en la figura anterior, las definiciones de estilo pueden ser compactas. Cada línea selecciona un elemento al que aplica y proporciona los valores de las propiedades. Las propiedades de un elemento se aplican como valores predeterminados para todos los demás elementos de HTML que contiene. Así, el estilo para body establece el estilo para los párrafos de texto en el cuerpo. También hay

convenientes métodos abreviados para los nombres de los colores (por ejemplo, red). El navegador llena con valores predeterminados cualquier parámetro de estilo que no se defina. Este comportamiento hace a las definiciones en la hoja de estilo opcionales; sin ellas puede haber una presentación razonable.

Podemos colocar las hojas de estilo en un archivo HTML (mediante la etiqueta <style>), pero es más común colocarlas en un archivo separado y hacer referencia a ellas. Por ejemplo, podemos modificar la etiqueta <head> de la página WASA para hacer referencia a una hoja de estilo en el archivo *estilowasa.css*, como se muestra en la figura 7-28. El ejemplo también muestra que el tipo MIME de los archivos CSS es *text/css*.

```
<head>
<title>WIDGETS AMALGAMADOS, S.A. </title>
<link rel=”stylesheet” type=”text/css” href= “awistyle.css” />
</head>
```

Figura 7-28. Inclusión de una hoja de estilo CSS.

Esta estrategia tiene dos ventajas. En primer lugar, permite aplicar un conjunto de estilos a muchas páginas en un sitio web. Esta organización otorga una apariencia consistente a las páginas, incluso aunque las hayan desarrollado distintos autores en distintos momentos, y permite modificar la apariencia de todo el sitio con sólo editar un solo archivo CSS y no el HTML. Podemos comparar este método con un archivo #include en un programa en C: al cambiar la definición de una macro ahí, se cambia en todos los archivos de programa que incluyen ese encabezado. La segunda ventaja es que los archivos de HTML que se descargan se mantienen en un tamaño pequeño. Esto se debe a que el navegador puede descargar una copia del archivo CSS para todas las páginas que lo referencien. No necesita descargar una nueva copia de las definiciones junto con cada página web.

7.3.3 Páginas web dinámicas y aplicaciones web

El modelo de página estático que hemos usado hasta ahora trata a las páginas como documentos multimedia que están vinculados de una manera conveniente. Fue un modelo adecuado en los primeros días de la web, cuando se pusieron en línea grandes cantidades de información. Hoy en día, la mayor parte de la emoción que rodea a la web está en usarla para aplicaciones y servicios. Ejemplos de ello son: comprar productos en sitios de comercio electrónico, buscar en catálogos de bibliotecas, explorar mapas, leer y enviar correo electrónico, y colaborar en documentos.

Estos nuevos usos son como el software de aplicación tradicional (por ejemplo, lectores de correo y procesadores de palabras). El detalle es que estas aplicaciones se ejecutan dentro del navegador, y los datos de usuario se almacenan en servidores en centros de datos de Internet. Utilizan protocolos web para acceder a la información a través de Internet, y el navegador para mostrar una interfaz de usuario. La ventaja de este método es que los usuarios no necesitan instalar programas de aplicación separados, y se puede acceder a los datos de usuario desde distintas computadoras, además de que el operador del servicio respalda la información. Está demostrando tener tanto éxito que rivaliza con el software de aplicación tradicional. Claro que también ayuda el hecho de que los grandes proveedores ofrezcan estas aplicaciones sin costo. Este modelo es la forma común de **computación en nube**, en la cual la computación pasa de las computadoras de escritorio individuales hacia grupos compartidos de servidores en Internet.

Para que actúen como aplicaciones, las páginas web ya no pueden ser estáticas. Es necesario el contenido dinámico. Por ejemplo, una página del catálogo de una biblioteca debe reflejar los libros disponibles en un momento dado, además de los libros que están prestados y por lo tanto no se encuentran disponibles.

Asimismo, una página útil de la Bolsa de Valores debe permitir al usuario interactuar con ella para ver los precios de las acciones durante distintos periodos y calcular tanto las ganancias como las pérdidas. Como lo sugieren estos ejemplos, el contenido dinámico se puede generar mediante programas que se ejecuten en el servidor o en el navegador (o en ambos lugares).

En esta sección examinaremos cada uno de estos casos en turno. La situación general es como se muestra en la figura 7-29. Por ejemplo, considere un servicio de mapas que permite al usuario introducir la dirección de una calle y presenta un mapa correspondiente de esa ubicación. Dada una solicitud para una ubicación, el servidor web debe usar un programa para crear una página que muestre el mapa para esa ubicación a partir de una base de datos de calles y demás información geográfica. Esta acción se muestra como los pasos 1 al 3. La solicitud (paso 1) provoca la ejecución de un programa en el servidor. El programa consulta una base de datos para generar la página apropiada (paso 2) y la devuelve al navegador (paso 3).

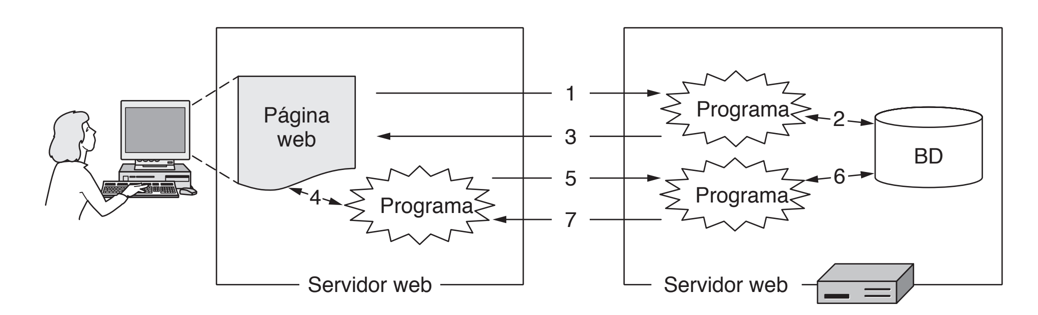


Figura 7-29. Páginas dinámicas.

Sin embargo, hay más sobre el contenido dinámico. La página que se devuelve puede a su vez contener programas que se ejecuten en el navegador. En nuestro ejemplo del mapa, el programa permitiría al usuario buscar rutas y explorar áreas cercanas con distintos niveles de detalle. Actualizaría la página, realizando acercamientos o alejamientos según las indicaciones del usuario (paso 4). Para manejar ciertas interacciones, el programa puede necesitar más datos del servidor. En este caso, enviará una solicitud al servidor (paso 5) para recuperar más información de la base de datos (paso 6) y devolver una respuesta (paso 7). Luego continuará actualizando la página (paso 4). Las solicitudes y respuestas se realizan en segundo plano; el usuario tal vez ni siquiera esté consciente de ellas debido a que por lo general, el URL de la página y el título no cambian. Al incluir programas del lado cliente, la página puede presentar una interfaz más receptiva que cuando se utilizan sólo programas del lado servidor.

Generación de páginas web dinámicas del lado servidor

Ahora veamos el caso de generación de contenido del lado servidor con más detalle. Una situación simple en la que se necesita el procesamiento del lado servidor es cuando se usan formularios. Considere la situación en que el usuario llena el formulario de pedidos WASA de la figura 7-25(b) y hace clic en el botón *Enviar pedido*. Cuando el usuario hace clic, se envía una solicitud al servidor en el URL especificado con el formulario (un *POST* a *http://widget.com/cgi-bin/pedido.cgi* en este caso), junto con el contenido del formulario que llenó el usuario. Hay que proporcionar estos datos a un programa o secuencia de comandos para que los procese. Así, el URL identifica el programa a ejecutar; los datos se proveen al programa en forma de entrada. En este caso, el procesamiento implicaría introducir el pedido en el sistema interno de WASA; actualizar los registros de los clientes y hacer el cargo a la tarjeta de crédito. La página devuelta

por esta solicitud dependerá de lo que ocurra durante el procesamiento. No es fija como una página estática. Si el pedido tiene éxito, la página devuelta podría proporcionar la fecha de entrega esperada. Si no tiene éxito, podría decir que los widgets solicitados están agotados o que la tarjeta de crédito no es válida por algún motivo.

La forma exacta en que el servidor ejecuta un programa en vez de recuperar un archivo depende del diseño del servidor web. Los protocolos web no lo especifican. Esto se debe a que la interfaz puede ser propietaria, por lo que el navegador no necesita conocer los detalles. En lo que a él respecta, simplemente está haciendo una solicitud para obtener una página.

Sin embargo, se han desarrollado varias API estándar para que los servidores web invoquen programas. La existencia de estas interfaces facilita a los desarrolladores el proceso de extender distintos servidores con aplicaciones web. A continuación analizaremos dos API para que el lector se dé una idea de lo que implican.

La primera API es un método para manejar solicitudes de páginas dinámicas, el cual ha estado disponible desde el inicio de la web. Se llama **CGI** (**Interfaz de Puerta de Enlace Común**, del inglés *Common Gateway Interface*) y se define en el RFC 3875. CGI provee una interfaz para permitir que los servidores web se comuniquen con programas de soporte y secuencias de comandos que pueden aceptar entradas (por ejemplo, de los formularios) y generar páginas HTML en respuesta. Estos programas pueden estar escritos en cualquier lenguaje que sea conveniente para el desarrollador, por lo general un lenguaje de secuencias de comandos para facilitar el avance. Puede elegir Python, Ruby, Perl o su lenguaje favorito.

Por convención, los programas que se invocan a través de CGI viven en un directorio llamado *cgi-bin*, el cual es visible en el URL. El servidor asocia una solicitud para este directorio con el nombre de un programa, y ejecuta ese programa en un proceso separado. Provee los datos enviados con la solicitud como entrada para el programa. La salida del programa produce una página web que se devuelve al navegador.

En nuestro ejemplo, el programa *pedido.cgi* se invoca con la entrada del formulario codificada, como se muestra en la figura 7-26. Este programa analiza los parámetros y procesa el pedido. Una convención útil es que devuelva el HTML para el formulario de pedido si no se provee un formulario de entrada. De esta manera, se asegurará de conocer la representación del formulario.

La segunda API que analizaremos es algo distinta. La metodología en este caso es incrustar pequeñas secuencias de comandos dentro de las páginas de HTML y hacer que las ejecute el mismo servidor para generar la página. Un lenguaje popular para escribir estas secuencias de comandos es **PHP** (**Preprocesador de Hipertexto**, del inglés *Hypertext Preprocessor*). Para usarlo, el servidor tiene que entender PHP, así como un navegador tiene que entender CSS para interpretar las páginas web con hojas de estilo. Por lo general, los servidores identifican las páginas web que contienen PHP con base en la extensión de archivo *php*, en vez de *html* o *htm*.

PHP es más simple de usar que CGI. Como ejemplo de la manera en que trabaja con los formularios, vea la figura 7-30(a). La parte superior de esta figura contiene una página HTML normal con un formulario simple en ella. Esta vez, la etiqueta <form> especifica que se debe invocar a *accion.php* para manejar los parámetros cuando el usuario envíe el formulario. La página despliega dos cuadros de texto, uno para solicitar el nombre y el otro para solicitar la edad. Después de que el usuario llena los dos cuadros y envía el formulario, el servidor analiza la cadena de texto que recibe de vuelta (la cual se parece a la de la figura 7-26) y después coloca el nombre en la variable *nombre* junto con la edad en la variable *edad*. Después empieza a procesar el archivo *accion.php*, que se muestra en la figura 7-30(b), como una respuesta. Durante el procesamiento de este archivo se ejecutan los comandos de PHP. Si el usuario escribió “Bárbara” y “24” en los cuadros, el archivo HTML que se envía de vuelta será el que se muestra en la figura 7-30(c). Por lo tanto, manejar formularios es un proceso que se vuelve en extremo simple mediante el uso de PHP.

```
<html>
<body>
<form action="accion.php" method="post">
<p> Por favor introduzca su nombre: <input type="text" name="nombre"> </p>
<p> Por favor introduzca su edad: <input type="text" name="edad"> </p>
<input type="submit">
</form>
</body>
</html>
```

(a)

```
<html>
<body>
<h1> Respuesta: </h1>
Hola <?php echo $nombre; ?>.
Predicción: el siguiente año tendrá <?php echo $edad + 1; ?>
</body>
</html>
```

(b)

```
<html>
<body>
<h1> Respuesta: </h1>
Hola Bárbara.
Predicción: el siguiente año tendrá 33
</body>
</html>
```

(c)

Figura 7-30. (a) Una página web que contiene un formulario. (b) Una secuencia de comandos de PHP para manejar la salida del formulario. (c) Salida de la secuencia de comandos de PHP cuando las entradas son “Bárbara” y “32”, respectivamente.

Aunque PHP es fácil de usar, en realidad es un poderoso lenguaje de programación diseñado para interactuar entre la web y una base de datos del servidor. Tiene variables, cadenas, arreglos y la mayoría de las estructuras de control que se encuentran en C, pero una E/S mucho más poderosa que la instrucción *printf*. PHP es código de fuente abierta, disponible de forma gratuita y muy popular. Se diseñó de manera específica para trabajar bien con Apache, que también es código de fuente abierta y es el servidor web más utilizado en el mundo. Para obtener más información sobre PHP, consulte a Valade (2009).

Ya vimos dos formas diferentes para generar páginas HTML dinámicas: las secuencias de comandos CGI y el PHP incrustado. Existen otras opciones disponibles. **JSP** (*JavaServer Pages*) es similar a PHP, excepto que la parte dinámica se escribe en el lenguaje de programación Java en lugar de PHP. Las páginas que utilizan esta técnica tienen la extensión de archivo *.jsp*. **ASP.NET** (*Active Server Pages .NET*) es la versión de Microsoft de PHP y JavaServer Pages. Utiliza programas escritos en el marco de trabajo (*framework*) de aplicaciones en red .NET propietario de Microsoft para generar el contenido dinámico. Las páginas que utilizan esta técnica tienen la extensión *.aspx*. Por lo general, la selección de una de estas tres técnicas tiene que ver más con políticas (código de fuente abierta contra Microsoft) que con la tecnología, puesto que estos tres lenguajes son más o menos similares.

Generación de páginas web dinámicas en el cliente

Las secuencias de comandos de CGI y PHP resuelven el problema de manejar la entrada y las interacciones con bases de datos en el servidor. Pueden aceptar información entrante de formularios, buscar

información en una o más bases de datos y generar páginas HTML con los resultados. Lo que ninguno de estos lenguajes puede hacer es responder a los movimientos del ratón o interactuar de manera directa con los usuarios. Para esto es necesario tener secuencias de comandos incrustadas en páginas HTML que se ejecuten en la máquina cliente y no en el servidor. Comenzando con HTML 4.0, tales secuencias de comandos son posibles mediante la etiqueta <script>. Las tecnologías utilizadas para producir estas páginas web interactivas se conocen en términos generales como **HTML dinámico**.

El lenguaje de secuencias de comandos más popular para el lado cliente es **JavaScript**, por lo que ahora le daremos un vistazo. A pesar de la similitud en los nombres, JavaScript no tiene casi nada que ver con el lenguaje de programación Java. Al igual que otros lenguajes de secuencias de comandos, es un lenguaje de muy alto nivel. Por ejemplo, en una sola línea de JavaScript es posible hacer que aparezca un cuadro de diálogo, esperar a que el usuario introduzca texto mediante el teclado y almacenar la cadena resultante en una variable. Las características de alto nivel como éstas hacen que JavaScript sea ideal para diseñar páginas web interactivas. Por otro lado, el hecho de que tiene más mutaciones que una mosca de fruta atrapada en una máquina de rayos X dificulta de manera considerable el proceso de escribir programas de JavaScript que funcionen en todas las plataformas, pero tal vez algún día se estabilice.

Como ejemplo de un programa de JavaScript, considere el de la figura 7-31. Al igual que el de la figura 7-30, despliega un formulario que pide un nombre y una edad, y después calcula cuántos años tendrá una persona el siguiente año. El cuerpo es casi el mismo que el del ejemplo de PHP; la diferencia principal es la declaración del botón *Submit* y su instrucción de asignación. Ésta indica al navegador que invoque la secuencia de comandos *respuesta* mediante un clic del botón y que la pase al formulario como un parámetro.

Lo que es completamente nuevo aquí es la declaración de la función *respuesta* de JavaScript en el encabezado del archivo HTML, un área que se reserva normalmente para títulos, colores de fondo, etc.

```
<html>
<head>
<script language=”javascript” type=”text/javascript”>
function respuesta(form_prueba) {
    var persona = form_prueba.nombre.value;
    var anios = eval(form_prueba.edad.value) + 1;
    document.open();
    document.writeln(“<html> <body>”);
    document.writeln(“Hola “ + persona + “.<br>”);
    document.writeln(“Predicción: el siguiente año tendrá “ + anios + “.”);
    document.writeln(“</body> </html>”);
    document.close();
}
</script>
</head>

<body>
<form>
Por favor introduzca su nombre: <input type=”text” name=”nombre”>
<p>
Por favor introduzca su edad: <input type=”text” name=”edad”> </p>
<p>
<input type=”button” value=”submit” onclick=”respuesta(this.form)”>
</form>
</body>
</html>
```

Figura 7-31. Uso de JavaScript para procesar un formulario.

Esta función extrae el valor del campo *nombre* del formulario y lo almacena como una cadena en la variable *persona*. También extrae el valor del campo *edad*, lo convierte en un entero mediante el uso de la función *eval*, le agrega 1 y almacena el resultado en *anios*. A continuación abre un documento para salida, utiliza el método *writeln* para realizar cuatro escrituras en dicho documento y, por último, lo cierra. Este documento es un archivo HTML, como podemos deducir de las diversas etiquetas HTML que contiene. A continuación el navegador despliega el documento en la pantalla.

Es muy importante tener en cuenta que, aunque PHP y JavaScript se ven similares en cuanto a que ambos incrustan código en archivos HTML, se procesan de una manera totalmente distinta. En el ejemplo de PHP de la figura 7-30, después de que el usuario hace clic en el botón *submit*, el navegador recolecta la información en una cadena larga y la envía al servidor como solicitud de una página de PHP. El servidor carga el archivo PHP y ejecuta la secuencia de comandos de PHP que está incrustada para producir una nueva página HTML. Esa página se devuelve al navegador para que la despliegue en pantalla. El navegador ni siquiera puede estar seguro de que fue producida por un programa. Este procesamiento se muestra como los pasos 1 a 4 en la figura 7-32(a).

En el ejemplo de JavaScript de la figura 7-31, al hacer clic en el botón *submit*, el navegador interpreta una función de JavaScript contenida en la página. Todo el trabajo se hace en forma local, dentro del navegador. No hay contacto con el servidor. Este procesamiento se muestra como los pasos 1 y 2 en la figura 7-32(b). Como consecuencia, el resultado se despliega casi al instante, mientras que con PHP puede haber un retraso de varios segundos antes de que llegue el HTML resultante al cliente.

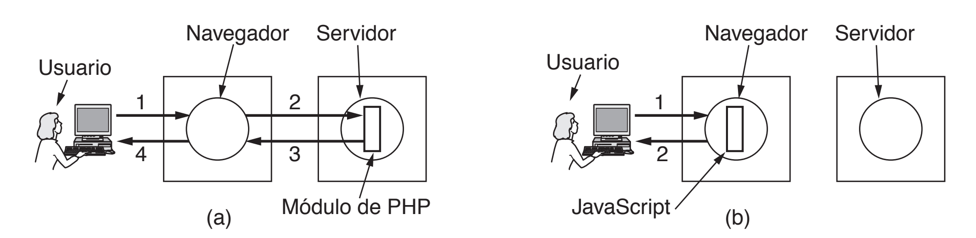


Figura 7-32. (a) Secuencias de comandos del lado servidor con PHP. (b) Secuencias de comandos del lado cliente con JavaScript.

Esta diferencia no significa que JavaScript sea mejor que PHP. Sus usos son completamente distintos. PHP (al igual que JSP y ASP) se utiliza cuando se requiere la interacción con una base de datos en el servidor. JavaScript (junto con otros lenguajes del lado cliente que mencionaremos, como VBScript) se utiliza cuando la interacción es con el usuario en la computadora cliente. Sin duda también es posible combinarlos, como veremos en breve.

JavaScript no es la única forma de crear páginas web muy interactivas. Una alternativa en las plataformas Windows es **VBScript**, que se basa en Visual Basic. Otro método popular entre plataformas es el uso de **applets**. Éstos son pequeños programas de Java compilados en instrucciones de máquina para una computadora virtual, conocida como **JVM** (**Máquina Virtual de Java**, del inglés *Java Virtual Machine*). Los applets se pueden incrustar en páginas HTML (entre <applet> y </applet>) y pueden ser interpretados por navegadores con soporte para JVM. Como los applets de Java se interpretan en vez de ejecutarse de manera directa, el intérprete de Java puede evitar que hagan "cosas malas". Por lo menos en teoría. En la práctica, los escritores de applets han descubierto un flujo casi interminable de errores en las bibliotecas de E/S de Java que pueden explotar.

La respuesta de Microsoft a los applets de Java de la empresa Sun fue permitir que las páginas web contuvieran **controles ActiveX**, que son programas compilados para lenguaje de máquina x86 y ejecutados en el hardware. Esta característica los hace muy rápidos y más flexibles que los subprogramas

interpretados de Java, porque pueden hacer cualquier cosa que un programa pueda hacer. Cuando Internet Explorer ve un control ActiveX en una página web, lo descarga, verifica su identidad y lo ejecuta. Sin embargo, la descarga y ejecución de programas extraños aumenta en forma considerable los problemas de seguridad, lo cual veremos en el capítulo 8.

Puesto que casi todos los navegadores pueden interpretar los programas de Java y JavaScript, un diseñador que desee crear una página web muy interactiva puede elegir por lo menos de entre dos técnicas, y si la portabilidad en múltiples plataformas no es importante, también puede elegir ActiveX. Como regla general, los programas de JavaScript son fáciles de escribir, los applets de Java se ejecutan más rápido y los controles ActiveX se ejecutan todavía más rápido. Además, puesto que todos los navegadores implementan exactamente la misma JVM pero no todos implementan la misma versión de JavaScript, los applets de Java son más portables que los programas de JavaScript. Para mayor información sobre JavaScript, hay muchos libros, cada uno con muchas páginas (por lo general, más de 1000). Por ejemplo, puede consultar a Flanagan (2010).

AJAX: JavaScript asíncrono y XML

Las aplicaciones web convincentes necesitan interfaces de usuario receptivas y un acceso transparente a los datos almacenados en servidores web remotos. Las secuencias de comandos en el cliente (por ejemplo, con JavaScript) y en el servidor (por ejemplo, con PHP) son tecnologías básicas que proporcionan algunas piezas de la solución. Esas tecnologías se utilizan frecuentemente con varias otras tecnologías clave en una combinación conocida como **AJAX** (**Javascript Asíncrono y Xml**, del inglés *Asynchronous JAvascript and Xml*). Muchas aplicaciones web llenas de funcionalidades, como Google Gmail, Maps y Docs, están escritas con AJAX.

El término AJAX es un poco confuso, ya que no es un lenguaje, sino un grupo de tecnologías que trabajan en conjunto para generar aplicaciones web que sean tan receptivas y poderosas como las aplicaciones de escritorio tradicionales. Estas tecnologías son:

1. HTML y CSS para presentar la información como páginas.
2. DOM (Modelo de Objetos de Documento) para modificar partes de las páginas mientras se despliegan en pantalla.
3. XML (Lenguaje de Marcado Extensible) para permitir que los programas intercambien datos de la aplicación con el servidor.
4. Una manera asíncrona para que los programas envíen y recuperen datos XML.
5. JavaScript como lenguaje para enlazar toda esta funcionalidad.

Como ésta es una colección bastante extensa, analizaremos cada una de las piezas para ver lo que contribuye. Ya hemos visto HTML y CSS. Son estándares para describir el contenido y la forma en que se debe desplegar en pantalla. Cualquier programa que pueda producir HTML y CSS, puede usar un navegador web como motor de visualización.

El **DOM** (**Modelo de Objetos de Documento**, del inglés *Document Object Model*) es la representación de una página HTML accesible para los programas. Esta representación se estructura como un árbol que refleja la estructura de los elementos de HTML. Por ejemplo, el árbol DOM del HTML de la figura 7-30(a) se proporciona en la figura 7-33. En la raíz hay un elemento *html* que representa a todo el bloque completo de HTML. Este elemento es el padre del elemento *body*, que a su vez es padre de un elemento *form*. El formulario tiene dos atributos que se arrastran al lado derecho, uno para el método del formulario (un *POST*) y otro para la acción del formulario (el URL a solicitar). Este elemento tiene tres hijos, los cuales reflejan las dos etiquetas de párrafo y una etiqueta de entrada que están contenidas dentro del formulario. En la parte inferior del árbol hay hojas que contienen elementos o literales, como las cadenas de texto.

La importancia del modelo DOM es que proporciona a los programas una forma directa de modificar partes de la página. Sin necesidad de volverla a escribir. Sólo hay que reemplazar el nodo que contiene la modificación. Al realizar esta modificación, el navegador actualizará de manera correspondiente la pantalla. Por ejemplo, si en DOM se modifica una imagen en una parte de la página, el navegador actualizará esa imagen sin modificar las demás partes de la página. Ya hemos visto a DOM en acción, con el ejemplo de JavaScript de la figura 7-31 que agrega líneas al elemento *document* para hacer que aparezcan nuevas líneas de texto en la parte inferior de la ventana del navegador. El DOM es un poderoso método para producir páginas que puedan evolucionar.

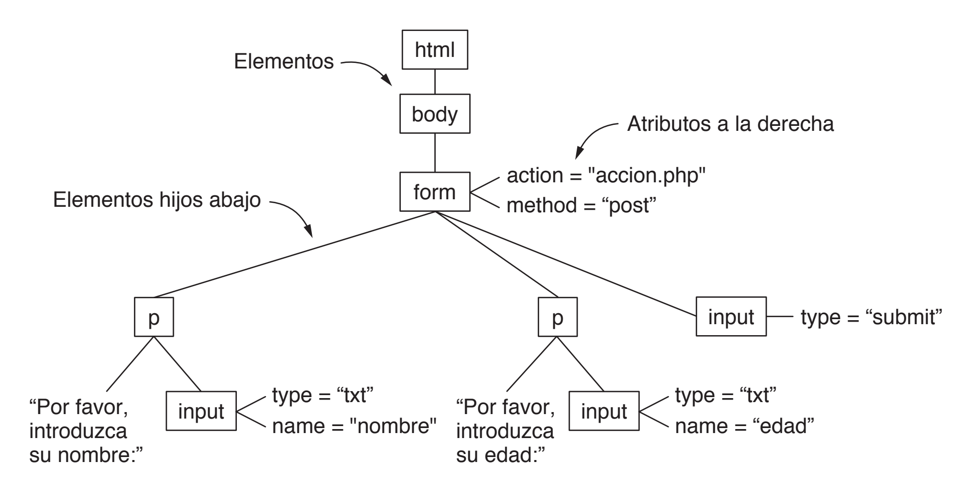


Figura 7-33. El árbol DOM para el HTML de la figura 7-30(a).

La tercera tecnología, **XML** (**Lenguaje de Marcado Extensible**, del inglés *eXtensible Markup Language*), es un lenguaje para especificar el contenido estructurado. HTML mezcla el contenido con el formato, puesto que se encarga de la presentación de la información. Sin embargo, a medida que se vuelven más comunes las aplicaciones web aumenta la necesidad de separar el contenido estructurado de su presentación. Por ejemplo, considere un programa que busca en la web el mejor precio de algún libro. Necesita analizar muchas páginas en busca del título y precio del artículo. Si las páginas web están en HTML, es muy difícil que un programa averigüe en dónde están el título y el precio.

Por esta razón, el W3C desarrolló el lenguaje XML (Bray y colaboradores, 2006) para permitir estructurar el contenido web para un procesamiento automatizado. A diferencia del HTML, en XML no hay etiquetas definidas. Cada usuario puede definir sus propias etiquetas. En la figura 7-34 se muestra un ejemplo simple de un documento XML, el cual define una estructura llamada lista_libros, que es una lista de libros. Cada uno tiene tres campos: título, autor y año de publicación. Estas estructuras son muy simples. Se permite tener estructuras con campos repetidos (por ejemplo, varios autores), campos opcionales (por ejemplo, el URL del libro de audio) y campos alternativos (por ejemplo, el URL de una librería si se sigue publicando o el URL de un libro de subastas si ya no se publica).

En este ejemplo, cada uno de los tres campos es una entidad indivisible, pero también es posible subdividir los campos todavía más. Por ejemplo, el campo del autor podría haberse organizado de la siguiente manera, para tener un control más detallado en cuanto a la búsqueda y el formato:

```
<autor>
    <primer_nombre> George </primer_nombre>
    <apellido_paterno>Zipf</apellido_paterno>
</autor>
```

Cada campo se puede subdividir en subcampos y subsubcampos, con una profundidad arbitraria.

```
<?xml version="1.0" ?>
<lista_libros>
<libro>
    <titulo> Human Behavior and the Principle of Least Effort </title>
    <autor> George Zipf </autor>
    <anio> 1949 </anio>
</libro>
<libro>
    <titulo> The Mathematical Theory of Communication </title>
    <autor> Claude E. Shannon </autor>
    <anio> 1949 </anio>
</libro>
<libro>
    <titulo> Nineteen Eighty-Four </title>
    <autor> George Orwell </autor>
    <anio> 1949 </anio>
</libro>
</lista_libro>
```

Figura 7-34. Un documento XML simple.

Todo lo que hace el archivo de la figura 7-34 es definir una lista que contiene tres libros. Es adecuada para transportar información entre programas que se ejecutan en navegadores y servidores, pero no dice nada sobre cómo desplegar el documento en pantalla como una página web. Para ello, un programa que consuma la información y juzgue que 1949 fue un excelente año para los libros podría producir un salida HTML donde los títulos se marquen como texto en cursiva. Como alternativa podemos usar un lenguaje llamado **XSLT** (**Transformaciones del Lenguaje de Hojas de Estilo eXtensible**, del inglés *eXtensible Stylesheet Language Transformations*) para definir cómo se debe transformar el XML en HTML. XSLT es igual que CSS, sólo que mucho más poderoso. A continuación le hablaremos sobre los detalles.

La otra ventaja de expresar los datos en XML en vez de usar HTML es que para los programas es más fácil de analizar. En un principio el HTML se escribía en forma manual (y aún se hace así en muchas ocasiones), por lo que cuando un documento contiene mucho HTML se vuelve algo desordenado. En ocasiones se omiten las etiquetas de cierre, como </p>. Otras no tienen una etiqueta de cierre asociada, como
. Además, hay otras etiquetas que se pueden anidar en forma inapropiada, y el caso de los nombres de etiquetas y atributos puede variar. La mayoría de los navegadores hacen su mejor esfuerzo para tratar de interpretar el documento lo más apegado posible. XML es más estricto y limpio en su definición. Los nombres de las etiquetas y los atributos siempre son en minúscula, las etiquetas siempre se deben cerrar en el orden inverso al orden en que se abrieron (o se debe indicar con claridad cuando se trata de una etiqueta vacía sin su correspondiente etiqueta de cierre), y los valores de los atributos deben ir entre comillas. Esta precisión hace que el análisis sea un proceso sencillo y sin ambigüedades.

Incluso el HTML se está definiendo en términos del XML. A esta metodología se le conoce como **XHTML** (**Lenguaje de Marcado de HiperTexto Extendido**, del inglés *eXtended HyperText Markup Language*). En esencia, es una versión "muy quisquillosa" de HTML. Las páginas de XHTML deben formarse de manera estricta a las reglas del XML, o de lo contrario el navegador no las aceptará. No más páginas web de mala calidad ni inconsistencias entre un navegador y otro. Al igual que con el XML,

la intención es producir páginas que los programas (en este caso, las aplicaciones web) puedan procesar mejor. Aunque el XHTML ha estado en funcionamiento desde 1998, ha tardado mucho en popularizarse. Las personas que producen HTML no ven la razón de necesitar XHTML, por lo que se ha retrasado el soporte en los navegadores. Ahora se está definiendo el HTML 5.0, de modo que una página se pueda representar como HTML o XHTML para ayudar en la transición. Eventualmente el XHMTL reemplazará al HTML, pero pasará mucho tiempo antes de que se complete esta transición.

XML también ha demostrado ser popular como lenguaje para la comunicación entre los programas. Cuando esta comunicación se lleva a cabo mediante el protocolo HTTP (que describiremos en la siguiente sección), se conoce como servicio web. En particular, **SOAP** (**Protocolo Simple de Acceso a Objetos**, del inglés *Simple Object Access Protocol*) es una forma de implementar servicios web en la que se realizan llamadas RPC entre los programas de una manera independiente del lenguaje y del sistema. El cliente sólo construye la solicitud como un mensaje XML y lo envía al servidor, mediante el protocolo HTTP. El servidor devuelve una respuesta en forma de un mensaje con formato de XML. De esta manera se pueden comunicar las aplicaciones en plataformas heterogéneas.

Ahora regresemos a AJAX. Nuestro razonamiento es simplemente que XML es un formato útil para intercambiar datos entre programas que se ejecutan en el navegador y el servidor. Sin embargo, para proveer una interfaz receptiva en el navegador mientras se envían o reciben datos, las secuencias de comandos deben ser capaces de realizar operaciones de **E/S asíncrona** para no bloquear la pantalla mientras esperan la respuesta de una solicitud. Por ejemplo, considere un mapa por el que nos podemos desplazar en el navegador. Al recibir la notificación de la acción de desplazamiento, la secuencia de comandos en la página del mapa puede solicitar más datos del mapa al servidor, si la vista del mapa está cerca del borde de los datos. La interfaz no se debe congelar mientras se obtienen esos datos. Dicha interfaz no ganaría premios de parte de los usuarios. En cambio, el desplazamiento debe continuar de manera uniforme. Al llegar los datos, se notifica a la secuencia de comandos para que los pueda utilizar. Si todo sale bien, se obtendrán los nuevos datos del mapa antes de que se requieran. Los navegadores modernos tienen soporte para este modelo de comunicación.

La pieza final del rompecabezas es un lenguaje de secuencias de comandos que mantenga unidos a todos los componentes de AJAX al proveer acceso a la lista anterior de tecnologías. En la mayoría de los casos este lenguaje es JavaScript, pero existen alternativas como VBScript. Anteriormente presentamos un ejemplo simple de JavaScript. Que no lo engañe esta sencillez. JavaScript tiene muchas peculiaridades, pero es un lenguaje de programación completo, con todo el poder de C o Java. Tiene variables, cadenas, arreglos, objetos, funciones y las estructuras de control usuales. También tiene interfaces específicas para el navegador y las páginas web. JavaScript puede rastrear el movimiento del ratón sobre los objetos en la pantalla, gracias a lo cual es fácil hacer que un menú aparezca de repente y conduzca a páginas web vívidas. Puede usar DOM para acceder a las páginas, manipular HTML y XML, y llevar a cabo una comunicación asíncrona mediante HTTP.

Antes de dejar el tema de las páginas dinámicas, sintetizaremos brevemente las tecnologías que hemos cubierto hasta ahora, para lo cual las relacionaremos en una sola figura. Se pueden generar páginas web completas al instante mediante diversas secuencias de comandos en la máquina servidor. Las secuencias de comandos se pueden escribir en lenguajes de extensión de servidor, como PHP, JSP o ASP.NET, o se pueden ejecutar como procesos CGI separados y, por ende, es posible escribirlos en cualquier lenguaje. Estas opciones se muestran en la figura 7-35.

Una vez que el navegador recibe estas páginas web, se tratan como páginas normales en HTML, CSS y otros tipos MIME, y simplemente se despliegan en pantalla. Se pueden instalar complementos que se ejecutan en el navegador y aplicaciones ayudantes que se ejecutan fuera del navegador para extender los tipos MIME soportados por el navegador.

También es posible generar contenido dinámico en el lado cliente. Los programas que están incrustados en páginas web se pueden escribir en JavaScript, VBScript, Java y otros lenguajes. Estos

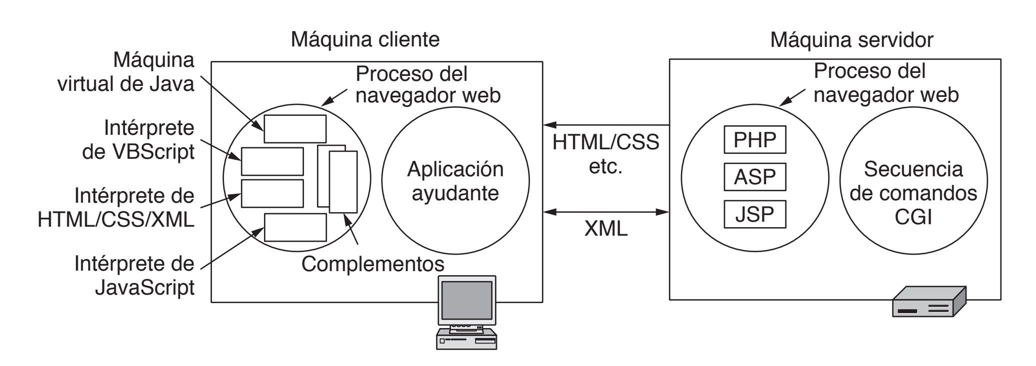


Figura 7-35. Varias tecnologías utilizadas para generar páginas dinámicas.

programas pueden realizar cálculos arbitrarios y actualizar la pantalla. Con AJAX, los programas en las páginas web pueden intercambiar XML y otros tipos de datos en forma asíncrona con el servidor. Este modelo soporta las aplicaciones web ricas en funcionalidad que se ven justo igual que las aplicaciones tradicionales, excepto que se ejecutan dentro del navegador y acceden a la información almacenada en servidores en Internet.

7.3.4 HTTP: el Protocolo de Transferencia de HiperTexto

Ahora que tenemos una idea sobre el contenido web y las aplicaciones, es tiempo de analizar el protocolo que se utiliza para transportar toda esta información entre servidores web y clientes: el **HTTP** (**Protocolo de Transferencia de HiperTexto**, del inglés *HyperText Transfer Protocol*), según lo especificado en el RFC 2616.

HTTP es un protocolo simple de solicitud-respuesta que por lo general opera sobre TCP. Especifica qué mensajes pueden enviar los clientes a los servidores, y qué respuestas reciben de estos mensajes. Los encabezados de solicitud y respuesta se proporcionan en ASCII, justo igual que en SMTP. El contenido se proporciona en un formato parecido a MIME, también como el SMTP. Este modelo simple fue en parte responsable del éxito anticipado de la web, ya que simplificó los procesos de desarrollo e implementación.

En esta sección analizaremos las propiedades más importantes del HTTP y su uso en la actualidad. Sin embargo, antes de entrar en detalles cabe mencionar que la forma en que se utiliza en Internet está evolucionando. HTTP es un protocolo de la capa de aplicación, debido a que opera sobre TCP y está muy asociado con la web. Ésta es la razón por la que lo vemos en este capítulo. Sin embargo, en otro sentido el HTTP se está volviendo más parecido a un protocolo de transporte que provee los medios para que los procesos se comuniquen el contenido entre un límite y otro de las distintas redes. Estos procesos no tienen que ser un navegador y un servidor web. Un reproductor de medios podría usar HTTP para comunicarse con un servidor y solicitar información sobre un álbum. El software antivirus podría usarlo para descargar las actualizaciones más recientes. Los desarrolladores podrían usarlo para obtener los archivos de sus proyectos. Los productos de electrónica para el consumidor, como los marcos de fotografías digitales, utilizan con frecuencia un servidor HTTP incrustado como interfaz para el mundo exterior. La comunicación de máquina a máquina se realiza a través de HTTP cada vez con más frecuencia. Por ejemplo, un servidor de una aerolínea podría usar SOAP (una RPC de XML sobre HTTP) para contactarse con un servidor de renta de autos y reservar un vehículo, todo como parte de un paquete vacacional. Es probable que estas tendencias continúen, junto con el uso expandido de HTTP.

Conexiones

La forma común en que un navegador contacta a un servidor es mediante el establecimiento de una conexión TCP por el puerto 80 en la máquina del servidor, aunque este procedimiento no se requiere formalmente. El valor de utilizar TCP es que ni los navegadores ni los servidores tienen que preocuparse por cómo manejar los mensajes largos, la confiabilidad o el control de la congestión. Todos estos aspectos se manejan mediante la implementación de TCP.

En los primeros días de la web con el HTTP 1.0, una vez que se establecía la conexión, se enviaba una solicitud y se obtenía una respuesta. Después se liberaba la conexión TCP. Este método era adecuado en un mundo en el que una página web típica consistía por completo de texto HTML. Sin embargo, la página web promedio creció con rapidez y contenía una gran cantidad de vínculos incrustados para contenido tal como iconos y otros elementos visuales. En consecuencia, establecer una conexión TCP para transportar cada icono por separado era una manera muy costosa de operar.

Esta observación condujo al HTTP 1.1, que soporta **conexiones persistentes**. Con ellas es posible establecer una conexión TCP, enviar una solicitud y obtener una respuesta, para después enviar solicitudes adicionales y obtener respuestas adicionales. A esta estrategia se le conoce también como **reutilización de la conexión**. Al amortizar los costos de establecimiento, inicio y liberación de TCP entre varias solicitudes, se reduce la sobrecarga relativa debido a TCP por cada solicitud. También es posible canalizar solicitudes; es decir, enviar la solicitud 2 antes de que haya llegado la respuesta a la solicitud 1.

En la figura 7-36 se muestra la diferencia en el desempeño entre estos tres casos. La parte (a) muestra tres solicitudes, una después de la otra y cada una en una conexión separada. Supongamos que esto representa una página web con dos imágenes incrustadas en el mismo servidor. Los URL de las imágenes se determinan al momento de obtener la página principal, por lo que se obtienen después de ésta. En la actualidad, una página ordinaria tiene alrededor de 40 objetos distintos que se deben obtener para presentarla, pero eso aumentaría mucho nuestros números, por lo cual sólo usaremos dos objetos incrustados.

En la figura 7-36(b), la página se obtiene mediante una conexión persistente. Esto es, la conexión TCP se abre al inicio, después se envían las mismas tres solicitudes, una después de la otra como antes,

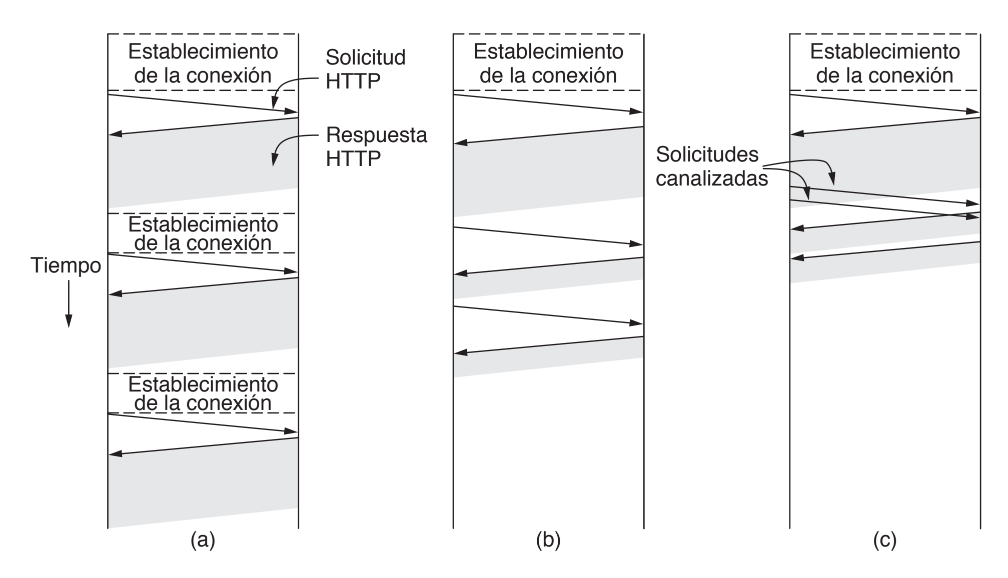


Figura 7-36. HTTP con (a) varias conexiones y solicitudes secuenciales. (b) Una conexión persistente y solicitudes secuenciales. (c) Una conexión persistente y solicitudes canalizadas.

y sólo entonces se cierra la conexión. Observe que la obtención se completa con más rapidez. Existen dos razones de la agilización. En primer lugar, no se desperdicia tiempo en establecer conexiones adicionales. Cada conexión TCP requiere cuando menos un tiempo de ida y vuelta para establecerse. En segundo lugar, la transferencia de la misma imagen se realiza con más rapidez. ¿Por qué es así? Se debe al control de la congestión de TCP. Al inicio de una conexión, TCP utiliza el procedimiento de inicio lento para incrementar la tasa de transferencia hasta que se aprende el comportamiento de la ruta de red. La consecuencia de este periodo de calentamiento es que varias conexiones TCP cortas tardan una mayor cantidad desproporcionada de tiempo en transferir información que una conexión TCP más larga.

Finalmente, en la figura 7-36(c) hay una conexión persistente y las solicitudes se canalizan. Específicamente, la segunda y la tercera se envían en rápida sucesión tan pronto como se haya recuperado lo suficiente de la página principal como para identificar que se deben obtener las imágenes. Con el transcurso del tiempo se envían las respuestas para estas solicitudes. Este método reduce el tiempo de inactividad del servidor, por lo que mejora el desempeño.

Sin embargo, las conexiones persistentes no son regaladas, puesto que surge un nuevo problema: cuándo cerrar la conexión. Una conexión a un servidor debe permanecer abierta mientras se carga la página. Entonces, ¿cuál es el problema? Hay una buena probabilidad de que el usuario haga clic en un vínculo que solicite otra página al servidor. Si la conexión permanece abierta, la siguiente solicitud se puede enviar de inmediato. Por lo tanto, no hay garantía de que el cliente vaya a hacer otra solicitud al servidor en cualquier momento. En la práctica, es común que los clientes y los servidores mantengan abiertas las conexiones persistentes hasta que hayan estado inactivos durante un intervalo corto (por ejemplo, 60 segundos), o cuando tengan una gran cantidad de conexiones abiertas y necesiten cerrar algunas.

El lector observador tal vez haya notado que hay una combinación de la que no hemos hablado hasta ahora. También es posible enviar una solicitud por cada conexión TCP, y ejecutar varias conexiones TCP en paralelo. Este método de **conexión paralela** se utilizaba mucho en los navegadores antes de las conexiones persistentes. Tiene la misma desventaja que las conexiones secuenciales: una sobrecarga adicional, pero a la vez ofrece un desempeño mucho mayor. Esto se debe a que al establecer y aumentar las conexiones en paralelo se oculta parte de la latencia. En nuestro ejemplo, las conexiones para las imágenes incrustadas se pueden establecer al mismo tiempo. Sin embargo, no se recomienda ejecutar muchas conexiones TCP con el mismo servidor. La razón es que TCP controla la congestión para cada conexión de manera independiente. Como consecuencia, las conexiones compiten entre sí, lo cual provoca una pérdida de paquetes adicional, y en conjunto son usuarios más agresivos de la red que una conexión individual. Las conexiones persistentes son superiores y es preferible usarlas en vez de las conexiones paralelas, ya que evitan una sobrecarga y no sufren de los problemas de congestión.

Métodos

Aunque HTTP se diseñó para utilizarlo en la web, se ha hecho intencionalmente más general de lo necesario con miras a futuros usos orientados a objetos. Por esta razón, se soportan otras operaciones, llamadas **métodos**, diferentes a las de solicitar una página web. Esta generalidad es lo que permite que SOAP exista.

Cada solicitud consiste en una o más líneas de texto ASCII; la primera palabra de la primera línea es el nombre del método solicitado. En la figura 7-37 se listan los métodos integrados. Los nombres son sensibles a mayúsculas y minúsculas, por lo que *GET* es un método válido y *get* no lo es.

El método *GET* solicita al servidor que envíe la página (cuando decimos “página” queremos decir “objeto”, en el caso más general, pero basta con pensar en una página como el contenido de un archivo

Método	Descripción
GET	Leer una página web.
HEAD	Leer el encabezado de una página web.
POST	Adjuntar a una página web.
PUT	Almacenar una página web.
DELETE	Eliminar la página web.
TRACE	Repetir la solicitud entrante
CONNECT	Conectarse a través de un proxy
OPTIONS	Consultar las opciones para una página

Figura 7-37. Los métodos de solicitud HTTP integrados.

para comprender los conceptos). La cual está codificada de forma adecuada en MIME. La gran mayoría de las solicitudes a servidores web son de tipo *GET*. La forma común de *GET* es:

```
GET nombredearchivo HTTP/1.1
```

en donde *nombredearchivo* nombra la página que se obtendrá y 1.1 es la versión del protocolo que se está utilizando.

El método *HEAD* sólo pide el encabezado del mensaje, sin la página real. Este método se puede utilizar para recolectar información para fines de indización, o sólo para evaluar la validez de un URL.

El método *POST* se utiliza para enviar formularios. Tanto este método como *GET* se emplean también para servicios web SOAP. Al igual que *GET*, porta también un URL, pero en lugar de sólo recuperar una página, envía datos al servidor (es decir, el contenido del formulario o los parámetros de RPC). Después el servidor hace algo con los datos que dependen del URL; conceptualmente adjunta los datos al objeto. El efecto podría ser comprar un elemento, por ejemplo, o llamar a un procedimiento. Finalmente, el método devuelve una página que indica el resultado.

El resto de los métodos no se utilizan mucho para navegar en la web. El método *PUT* es lo inverso a *GET*: en lugar de leer la página, la escribe. Este método hace posible la construcción de una colección de páginas web en un servidor remoto. El cuerpo de la solicitud contiene la página. Se puede codificar mediante el uso de MIME, en cuyo caso las líneas que siguen al método *PUT* podrían incluir encabezados de autentificación, para demostrar que el invocador realmente tiene permiso de realizar la operación solicitada.

DELETE hace lo que usted podría esperar: elimina la página, o al menos indica que el servidor web está de acuerdo con eliminar la página. Al igual que con *PUT*, la autentificación y el permiso juegan un papel importante aquí.

El método *TRACE* es para la depuración. Indica al servidor que regrese la solicitud. Este método es útil cuando las solicitudes no se están procesando de manera correcta y el cliente desea saber cuál solicitud ha obtenido realmente el servidor.

El método *CONNECT* permite a un usuario realizar una conexión con un servidor web por medio de un dispositivo intermedio, como una caché web.

El método *OPTIONS* proporciona una forma para que el cliente consulte al servidor sobre una página y obtenga los métodos y encabezados que se pueden usar con esa página.

Cada solicitud obtiene una respuesta que consiste en una línea de estado, y posiblemente de información adicional (por ejemplo, toda o parte de una página web). La línea de estado contiene un código de

estado de tres dígitos que indica si se atendió la solicitud, y si no, por qué. El primer dígito se utiliza para dividir las respuestas en cinco grupos principales, como se muestra en la figura 7-38. En la práctica, los códigos 1xx no se utilizan con frecuencia. Los códigos 2xx indican que la solicitud se manejó de manera exitosa y que se regresa el contenido (si hay alguno). Los códigos 3xx indican al cliente que busque en otro lado, ya sea mediante el uso de un URL diferente o en su propia caché (lo cual veremos más adelante). Los códigos 4xx significan que la solicitud falló debido a un error del cliente, por ejemplo: una solicitud inválida o una página inexistente. Por último, los errores 5xx indican que el servidor tiene un problema interno, debido a un error en su código o a una sobrecarga temporal.

Código	Significado	Ejemplos
1xx	Información	100 = el servidor acepta manejar la solicitud del cliente.
2xx	Éxito	200 = la solicitud es exitosa; 204 = no hay contenido.
3xx	Redirección	301 = se movió la página; 304 = la página en caché aún es válida.
4xx	Error del cliente	403 = página prohibida; 404 = no se encontró la página.
5xx	Error del servidor	500 = error interno del servidor; 503 = intentar más tarde.

Figura 7-38. Los grupos de respuesta del código de estado.

Encabezados de mensaje

A la línea de solicitud (por ejemplo, la línea con el método *GET*) le pueden seguir líneas adicionales que contienen más información. Éstas se llaman **encabezados de solicitud**. Podemos comparar esta información con los parámetros de la llamada a un procedimiento. Las respuestas también pueden tener **encabezados de respuesta**. Algunos encabezados se pueden utilizar en cualquier dirección. En la figura 7-39 se muestra una selección de los más importantes. Esta lista no es corta, por lo que como podría imaginar, es común tener una variedad de encabezados en cada solicitud y respuesta.

El encabezado *User-Agent* permite que el cliente informe al servidor sobre la implementación de su navegador (por ejemplo, *Mozilla/5.0* y *Chrome/5.0.375.125*). Esta información es útil para que los servidores puedan ajustar sus respuestas para el navegador, ya que los distintos navegadores pueden tener herramientas y comportamientos que varíen de manera considerable.

Los cuatro encabezados *Accept* indican al servidor lo que el cliente está dispuesto a aceptar en caso de que tenga un repertorio limitado de lo que es aceptable. El primer encabezado especifica los tipos MIME aceptados (por ejemplo, *text/html*). El segundo proporciona el conjunto de caracteres (por ejemplo, *ISO-8859-5* o *Unicode-1-1*). El tercero tiene que ver con métodos de compresión (por ejemplo, *gzip*). El cuarto indica un idioma natural (por ejemplo, español). Si el servidor tiene varias páginas a elegir, puede utilizar esta información para proporcionar la que el cliente esté buscando. Si es incapaz de satisfacer la solicitud, se regresa un código de error y la solicitud falla.

Los encabezados *If-Modified-Since* e *If-None-Match* se utilizan con la caché. Permiten al cliente pedir que se envíe una página sólo si la copia en la caché ya no es válida. Más adelante hablaremos sobre el uso de la caché.

El encabezado *Host* da nombre al servidor. Este encabezado se toma del URL y es obligatorio. Se utiliza porque algunas direcciones IP pueden proporcionar varios nombres DNS y el servidor necesita una forma para saber a qué host debe entregar la solicitud.

El encabezado *Authorization* es necesario para páginas que están protegidas. En este caso, tal vez el cliente debe probar que tiene permiso para ver la página solicitada. Este encabezado se utiliza para ese caso.

Encabezado	Tipo	Contenidos
User-Agent	Solicitud	Información sobre el navegador y su plataforma.
Accept	Solicitud	El tipo de páginas que puede manejar el cliente.
Accept-Charset	Solicitud	Los conjuntos de caracteres que son aceptables para el cliente.
Accept-Encoding	Solicitud	Las codificaciones de página que puede manejar el cliente.
Accept-Language	Solicitud	Los idiomas naturales que puede manejar el cliente.
If-Modified-Since	Solicitud	Hora y fecha para verificar la actualidad de un mensaje.
If-None-Match	Solicitud	Etiquetas enviadas previamente para verificar la actualidad de un mensaje.
Host	Solicitud	El nombre DNS del servidor.
Authorization	Solicitud	Una lista de las credenciales del cliente.
Referer	Solicitud	El URL anterior desde el cual provino la solicitud.
Cookie	Solicitud	La cookie establecida previamente que se regresa al servidor.
Set-Cookie	Respuesta	La cookie que debe guardar el cliente.
Server	Respuesta	Información sobre el servidor.
Content-Encoding	Respuesta	Cómo se codifica el contenido (por ejemplo, gzip).
Content-Language	Respuesta	El lenguaje natural utilizado en la página.
Content-Length	Respuesta	La longitud de la página en bytes.
Content-Type	Respuesta	El tipo MIME de la página.
Content-Range	Respuesta	Identifica una parte del contenido de la página.
Last-Modified	Respuesta	Hora y fecha de la última modificación de la página.
Expires	Respuesta	Hora y fecha en que la página dejará de ser válida.
Location	Respuesta	Indica al cliente a dónde enviar su solicitud.
Accept-Ranges	Respuesta	Indica que el servidor aceptará solicitudes de rango de bytes.
Date	Ambas	Fecha y hora en que se envió el mensaje.
Range	Ambas	Identifica una parte de una página.
Cache-Control	Ambas	Directivas para manejar las cachés.
ETag	Ambas	Etiqueta para el contenido de la página.
Upgrade	Ambas	El protocolo al que el emisor desea conmutar.

Figura 7-39. Algunos encabezados de mensaje HTTP.

El cliente usa el encabezado *Referer* mal escrito para proporcionar el URL que hizo referencia al URL que ahora se solicita. Lo más común es que sea el URL de la página anterior. Este encabezado es particularmente útil para rastrear la navegación web, ya que indica a los servidores cómo llegó un cliente a la página.

Aunque las cookies se tratan en el RFC 2109 en lugar de en el RFC 2616, también tienen encabezados. El encabezado *Set-Cookie* es la forma en que los servidores envían cookies a los clientes. Éste debe

guardar la cookie y regresarla en las solicitudes posteriores al servidor, mediante el uso del encabezado *Cookie* (cabe mencionar que hay una especificación más reciente para las cookies con encabezados más recientes, conocida como RFC 2965, pero la industria la rechazó en gran parte, por lo cual no se implementa mucho).

En las respuestas se usan muchos otros encabezados. El encabezado *Server* permite que el servidor identifique su versión de compilación de software si lo desea. Los siguientes cinco encabezados, que empiezan todos con *Content-*, permiten al servidor describir las propiedades de la página que va a enviar.

El encabezado *Last-Modified* indica cuándo se modificó la página por última vez y *Expires* indica por cuánto tiempo será válida la página. Ambos encabezados desempeñan un papel importante en el uso de la caché para las páginas.

El servidor utiliza el encabezado *Location* para informar al cliente que debe probar con un URL distinto. Esto se puede usar si se cambió la ubicación de la página o para permitir que varios URL hagan referencia a la misma página (posiblemente en distintos servidores). También se utiliza para las empresas que tienen una página web principal en el dominio *com*, pero redirigen a sus clientes a una página nacional o regional con base en sus direcciones IP o su idioma preferido.

Si una página es muy grande, tal vez un cliente pequeño no quiera recibirla toda de una sola vez. Algunos servidores aceptan solicitudes de rangos de bytes, de modo que la página se pueda obtener en varias unidades pequeñas. El encabezado *Accept-Ranges* anuncia la disposición del servidor para manejar este tipo de solicitud parcial de página.

Ahora veamos los encabezados que se pueden usar en ambas direcciones. *Date* se puede usar en ambas direcciones y contiene la fecha y hora en que se envió el mensaje, mientras que *Range* indica el rango de bytes de la página que se provee mediante la respuesta.

El encabezado *ETag* proporciona una etiqueta corta que sirve como nombre para el contenido de la página. Se utiliza para trabajar con la caché. El encabezado *Cache-Control* proporciona otras instrucciones explícitas sobre cómo colocar páginas en la caché (o, lo que es más común, cómo no colocarlas).

Por último, el encabezado *Upgrade* se utiliza para conmutar a un nuevo protocolo de comunicación, como un protocolo HTTP futuro o un transporte seguro. Este encabezado permite al cliente anunciar lo que puede soportar, y al servidor afirmar lo que está usando.

Almacenamiento en caché

A menudo las personas regresan a las páginas web que han visto antes, y es común que las páginas web relacionadas tengan los mismos recursos incrustados. Algunos ejemplos son las imágenes que se utilizan para navegar a través del sitio, así como las hojas de estilo y las secuencias de comandos comunes. Sería un desperdicio obtener todos estos recursos para esas páginas cada vez que se desplieguen en pantalla, puesto que el navegador ya tiene una copia.

Al proceso de guardar y ocultar las páginas que se obtienen para usarlas después se le conoce como **almacenamiento en caché**. La ventaja es que, cuando se puede reutilizar una página en caché, no es necesario repetir la transferencia. HTTP tiene soporte integrado para ayudar a los clientes a identificar cuándo pueden reutilizar páginas. Este soporte mejora el desempeño al reducir tanto el tráfico como la latencia en la red. La desventaja es que el navegador ahora debe guardar páginas, pero esto casi siempre vale la pena, ya que el almacenamiento local no es muy costoso. Por lo general las páginas se mantienen en el disco, de modo que se puedan usar al ejecutar el navegador en una fecha posterior.

El verdadero problema con el almacenamiento en caché en HTTP es cómo determinar que una copia de una página previamente colocada en caché es la misma que se obtendría del servidor otra vez. No podemos hacer esta determinación únicamente con base en el URL. Por ejemplo, tal vez el URL

puede proporcionar una página que muestra la noticia más reciente. El contenido de esta página se actualizará con frecuencia, incluso cuando el URL permanezca igual. O por el contrario, el contenido de la página puede ser una lista de los dioses de la mitología griega y romana. Lo más seguro es que esta página cambie con menos rapidez.

HTTP utiliza dos estrategias para lidiar con este problema, las cuales se muestran en la figura 7-40 como formas de procesamiento entre la solicitud (paso 1) y la respuesta (paso 5). La primera estrategia es la validación de páginas (paso 2). Se consulta la caché y, si tiene una copia de una página para el URL solicitado que se sabe está actualizada (es decir, aún es válida), no hay necesidad de obtenerla de nuevo del servidor. En cambio, se puede regresar la página en caché directamente. Para realizar esta determinación podemos utilizar el encabezado *Expires* que se devolvió cuando se obtuvo por primera vez la página en caché, junto con la fecha y hora actuales.

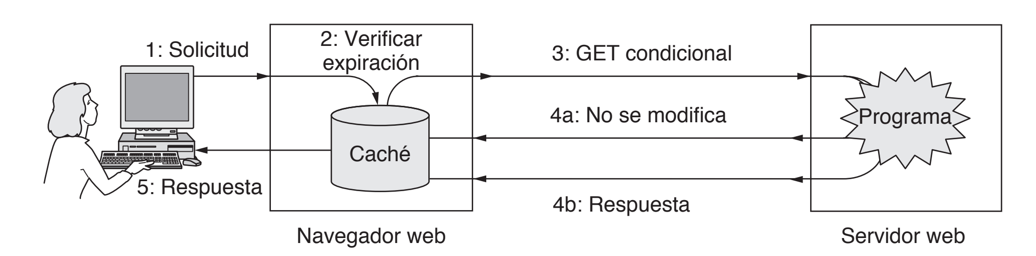


Figura 7-40. Almacenamiento en caché en HTTP.

Sin embargo, no todas las páginas incluyen un encabezado *Expires* apropiado que les indica cuándo debe obtener la página nuevamente. Después de todo, es difícil hacer predicciones —en especial sobre el futuro. En este caso, el navegador puede usar la heurística. Por ejemplo, si la página no se ha modificado en el último año (según lo que indica el encabezado *Last-Modified*), es de suponer que no cambiará en la siguiente hora. Por lo tanto, no hay garantía, por lo que ésta podría ser una mala suposición. Por ejemplo, el mercado de valores podría haber cerrado sus operaciones del día, de modo que la página no cambiará durante horas, pero cambiará con rapidez una vez que inicie la siguiente sesión de compra-venta de valores. Por ende, la factibilidad de almacenar una página en caché variará de manera considerable con el tiempo. Por esta razón se debe usar la heurística con cuidado, aunque por lo general funciona bien en la práctica.

Encontrar páginas que no hayan expirado es el uso más benéfico del almacenamiento en caché, ya que significa que no hay que tener ningún contacto con el servidor. Por desgracia, esto no siempre funciona. Los servidores deben usar el encabezado *Expires* de manera conservadora, ya que tal vez no estén seguros acerca de cuándo se actualizará una página. Por lo tanto, las copias en caché tal vez estén actualizadas pero el cliente no lo sepa.

En este caso se utiliza la segunda estrategia: preguntar al servidor si la copia en caché sigue siendo válida. Esta solicitud es un **GET condicional** y se muestra en la figura 7-40 como el paso 3. Si el servidor sabe que la copia en caché aún es válida, puede enviar una respuesta corta para decirlo (paso 4a). En caso contrario, debe enviar la respuesta completa (paso 4b).

Se utilizan más campos de encabezado para que el servidor pueda verificar si una copia en caché aún es válida. El cliente tiene la hora en que se actualizó una página en caché por última vez, a través del encabezado *Last-Modified*. Puede enviar esta hora al servidor mediante el uso del encabezado *If-Modified-Since* para pedir la página sólo si se modificó en ese tiempo.

Como alternativa, el servidor puede devolver un encabezado *ETag* con una página. Este encabezado proporciona una etiqueta que es un nombre corto para el contenido de la página, como una suma de

verificación, sólo que mejor (puede ser un *hash* criptográfico, el cual describiremos en el capítulo 8). Para validar las copias en caché, el cliente envía un encabezado *If-None-Match* con una lista de las etiquetas. Si alguna de ellas coincide con el contenido con el que el servidor respondería, se puede usar la correspondiente copia en caché. Podemos utilizar este método cuando no sea conveniente ni útil determinar la actualidad de una página. Por ejemplo, un servidor puede regresar distinto contenido para el mismo URL, dependiendo de los idiomas y tipos MIME que se prefieran. En este caso, la fecha de modificación por sí sola no ayudará al servidor a determinar si la página en caché es actual.

Por último, observe que ambas estrategias de almacenamiento en caché son anuladas por las directivas que se transportan en el encabezado *Cache-Control*. Estas directivas se pueden usar para restringir el almacenamiento en caché (por ejemplo, *no-cache*) cuando no sea apropiado. Un ejemplo es una página dinámica, que será diferente la próxima vez que se obtenga. Las páginas que requieren autorización tampoco se almacenan en caché.

Hay mucho más en cuanto al almacenamiento en caché, pero sólo tenemos el espacio suficiente para expresar dos puntos importantes. En primer lugar, el almacenamiento en caché se puede realizar en otros lugares además del navegador. En el caso general, las solicitudes HTTP se pueden enrutar a través de una serie de cachés. El uso de una caché externa al navegador se conoce como **almacenamiento en caché proxy**. Cada nivel agregado de almacenamiento en caché puede ayudar a reducir las solicitudes en niveles superiores de la cadena. Es común que las organizaciones como los ISP y las empresas ejecuten cachés proxy para obtener los beneficios de almacenar páginas en caché entre distintos usuarios. En la sección 7.5, al final de este capítulo, analizaremos el almacenamiento en caché proxy con el tema más amplio de la distribución de contenido.

En segundo lugar, las cachés proveen un aumento importante en el desempeño, pero no tanto como podríamos esperar. La razón es que, aunque hay ciertos documentos populares en la web, también hay muchos documentos no tan populares que las personas obtienen, muchos de los cuales son también muy extensos (por ejemplo, los videos). La "cola larga" de los documentos no populares ocupa espacio en las cachés, y la cantidad de solicitudes que se pueden manejar desde la caché aumenta de manera lenta con el tamaño de la caché. Siempre existe la probabilidad de que las cachés web puedan manejar menos de la mitad de las solicitudes. Consulte a Breslau y colaboradores (1999) para obtener más información.

Experimentación con el HTTP

Puesto que el HTTP es un protocolo ASCII, es bastante fácil para una persona en una terminal (lo opuesto a un navegador) comunicarse de manera directa con los servidores web. Todo lo que se necesita es una conexión TCP al puerto 80 en el servidor. Se anima a los lectores a experimentar con la siguiente secuencia de comandos. Funcionará en la mayoría de los *shells* de UNIX y en la ventana de comandos de Windows (una vez que se habilite el programa telnet).

```
telnet www.ietf.org 80
GET /rfc.html HTTP/1.1
Host: www.ietf.org
```

Esta secuencia de comandos inicia una conexión telnet (es decir, TCP) al puerto 80 en el servidor web de la IETF, *www.ietf.org*. Después viene el comando *GET* que nombra la ruta del URL y el protocolo. Pruebe con servidores y direcciones URL de su elección. La siguiente línea es el encabezado *Host* obligatorio. Una línea en blanco después del último encabezado es obligatoria, ya que indica al servidor que no hay más encabezados de solicitud. A continuación el servidor enviará la respuesta. Dependiendo del servidor y el URL, se pueden observar muchos tipos distintos de encabezados y páginas.

7.3.5 La web móvil

La Web se utiliza desde casi cualquier tipo de computadora, incluyendo los teléfonos móviles. Puede ser muy útil navegar por la web a través de una red inalámbrica mientras nos desplazamos de un lugar a otro. También se presentan problemas técnicos, ya que gran parte del contenido web está diseñado para presentaciones ostentosas en computadoras de escritorio con conectividad de banda ancha. En esta sección describiremos cómo se está desarrollando el acceso web desde dispositivos móviles, mejor conocido como **web móvil**.

En comparación con las computadoras de escritorio en el trabajo o el hogar, los teléfonos móviles presentan varias dificultades para la navegación web:

1. Las pantallas relativamente pequeñas impiden ver páginas extensas e imágenes grandes.
2. Las capacidades limitadas de entrada hacen que sea tedioso introducir direcciones URL u otro tipo de entrada extensa.
3. El ancho de banda de red está limitado a través de los enlaces inalámbricos, en especial en las redes celulares (3G), en donde con frecuencia también es costoso.
4. La conectividad puede ser intermitente.
5. El poder de cómputo está limitado por cuestiones de vida de la batería, tamaño, disipación de calor y costo.

Estas dificultades significan que hay una gran probabilidad de que el simple uso de contenido de escritorio para la web móvil provoque una experiencia frustrante en el usuario.

En las primeras metodologías para la web móvil se ideó una nueva pila de protocolos enfocada a los dispositivos inalámbricos con capacidades limitadas. **WAP** (**Protocolo de Aplicaciones Inalámbricas**, del inglés *Wireless Application Protocol*) es el ejemplo más conocido de esta estrategia. El esfuerzo de WAP lo empezaron en 1997 los principales distribuidores de teléfonos móviles: Nokia, Ericsson y Motorola. Sin embargo, ocurrió algo inesperado en el camino. En la siguiente década, el ancho de banda de red y las capacidades de los dispositivos aumentaron de manera considerable gracias a la implementación de los servicios de datos 3G y los teléfonos móviles con pantallas más grandes a color, procesadores más rápidos y capacidades inalámbricas 802.11. De repente los teléfonos móviles ya eran capaces de ejecutar navegadores web simples. Aún hay un vacío entre estos teléfonos móviles y los equipos de escritorio que nunca se cerrará, pero han desaparecido muchos de los problemas tecnológicos que dieron ímpetu al tema de una pila de protocolos separada.

El método que se utiliza cada vez con más frecuencia es ejecutar los mismos protocolos web para los equipos móviles y los de escritorio, y hacer que los sitios web ofrezcan contenido amigable para los móviles cuando el usuario se encuentre en un dispositivo de este tipo. Los servidores web son capaces de detectar si deben regresar versiones de escritorio o móviles de las páginas web con sólo analizar los encabezados de la solicitud. El encabezado *User-Agent* es en especial útil en esta cuestión, ya que identifica el software del navegador. Así, cuando un servidor web recibe una solicitud, puede analizar los encabezados y regresar una página con imágenes pequeñas, menos texto y una navegación más simple para un iPhone, y una página completa al usuario en una computadora portátil.

El W3C está fomentando esta metodología de varias formas. Una de ellas es estandarizar las mejores prácticas para el contenido web móvil. En la primera especificación se provee una lista de 60 de esas mejores prácticas (Rabin y McCathieNevile, 2008). La mayoría de estas prácticas llevan a cabo pasos prudentes para reducir el tamaño de las páginas, incluyendo el uso de la compresión, ya que los costos de comunicación son más altos que los del poder de cómputo; también se maximiza la efectividad del almacenamiento en caché. Esta metodología anima a los sitios, en especial los grandes, a que creen versiones web móviles de su contenido, ya que es todo lo que necesitan para capturar a los usuarios de la web móvil.

Para ayudar a esos usuarios en el proceso, también existe un logo que indica las páginas que se pueden ver (bien) en la web móvil.

Otra herramienta útil es una versión simplificada de HTML, conocida como **XHTML Básico**. Este lenguaje es un subconjunto de XHTML destinado para usarse en teléfonos móviles, televisiones, dispositivos PDA, máquinas expendedoras, radiolocalizadores, autos, máquinas de juegos e incluso hasta relojes. Por esta razón no soporta las hojas de estilo, las secuencias de comandos ni los marcos, pero la mayoría de las etiquetas estándar están ahí. Se agrupan en 11 módulos. Algunas son requeridas; otras son opcionales. Todas están definidas en XML. En la figura 7-41 se listan los módulos y algunas etiquetas de ejemplo.

Módulo	¿Requerido?	Función	Etiquetas de ejemplo
Estructura	Sí	Estructura del documento.	body, head, html, title
Texto	Sí	Información.	br, code, dfn, em, hn, kbd, p, strong
Hipertexto	Sí	Hipervínculos.	a
Lista	Sí	Listas detalladas.	dl, dt, dd, ol, ul, li
Formularios	No	Formularios de entrada de datos.	form, input, label, option, textarea
Tablas	No	Tablas rectangulares.	caption, table, td, th, tr
Imagen	No	Imágenes.	Img
Objeto	No	Applets, mapas, etcétera.	object, param
Metainformación	No	Información extra.	meta
Vínculo	No	Similar a <a>.	link
Base	No	Punto de inicio del URL.	base

Figura 7-41. Los módulos y etiquetas de XHTML Básico.

Sin embargo, no todas las páginas estarán diseñadas para trabajar bien en la web móvil. Por lo tanto, una metodología complementaria es el uso de la **transformación de contenido** o **transcodificación**. En esta metodología, una computadora que está entre el móvil y el servidor recibe las solicitudes del móvil, obtiene el contenido del servidor y lo transforma en contenido web móvil. Una transformación simple es reducir el tamaño de las imágenes grandes, para lo cual se ajusta el formato para producir una resolución más baja. Se pueden usar muchas otras transformaciones pequeñas pero útiles. La transcodificación se ha utilizado con cierto éxito desde los primeros días de la web móvil. Consulte, por ejemplo, a Fox y colaboradores (1996). Sin embargo, cuando se utilizan ambas metodologías hay una tensión entre las decisiones de contenido móvil que toma el servidor y las que toma el transcodificador. Por ejemplo, tal vez un sitio web seleccione una combinación específica de imagen y texto para un usuario de la web móvil, sólo para que un transcodificador cambie el formato de la imagen.

Hasta ahora nuestra discusión ha sido sobre el contenido, no los protocolos, ya que éste es el mayor problema para realizar la web móvil. Sin embargo, mencionaremos con brevedad la cuestión de los protocolos. En la web, el uso de los protocolos HTTP, TCP e IP puede consumir una cantidad considerable de ancho de banda debido a las sobrecargas de los protocolos, como los encabezados. Para lidiar con este problema, WAP y otras soluciones definieron protocolos de propósito especial. Esto resulta ser en gran parte innecesario. Las tecnologías de compresión de encabezados, como la **ROHC**

(**Compresión RObusta de Encabezados**, del inglés *RObust Header Compression*) que se describe en el capítulo 6, pueden reducir las sobrecargas de estos protocolos. De esta forma, es posible tener un conjunto de protocolos (HTTP, TCP, IP) y utilizarlos a través de enlaces de ancho de banda alto y bajo. El uso a través de los enlaces de bajo ancho de banda simplemente requiere la activación de la compresión de los encabezados.

7.3.6 Búsqueda web

Para terminar nuestra descripción de la web, analizaremos lo que sin duda es la aplicación web más exitosa: la búsqueda. En 1998, Sergey Brin y Larry Page, que entonces eran estudiantes de maestría en Stanford, formaron una empresa nueva llamada Google para construir un mejor motor de búsqueda web. Estaban armados con la entonces radical idea de que un algoritmo de búsqueda que contara cuántas veces una página era apuntada por otras representaba una mejor medida de su importancia, en vez de contar cuántas veces contenía las palabras clave que se estaban buscando. Por ejemplo, muchas páginas tienen vínculos hacia la página principal de Cisco, lo cual hace a ésta más importante para un usuario que busca "Cisco" que una página que no esté relacionada con la empresa y simplemente utilice la palabra "Cisco" muchas veces.

Ellos tenían razón. Se demostró que era posible construir un mejor motor de búsqueda y las personas se apresuraron a usarlo. Gracias al respaldo del capital de riesgo, Google creció de manera considerable. Se convirtió en empresa pública en 2004, con una capitalización de mercado de $23 mil millones. Para 2010 se estimó que ejecutaba más de un millón de servidores en centros de datos esparcidos por todo el mundo.

En cierto sentido, la búsqueda es simplemente otra aplicación web, aunque es una de las más maduras debido a que ha estado en desarrollo desde los primeros días de la web. Sin embargo, la búsqueda web ha demostrado ser indispensable en el uso diario. Se estima que se realizan más de mil millones de búsquedas en la web cada día. Las personas que buscan todo tipo de información usan la búsqueda como punto inicial. Por ejemplo, para averiguar en dónde comprar Vegemite en Seattle, no hay un sitio web obvio que podamos usar como punto de partida. Pero es probable que un motor de búsqueda conozca una página con la información deseada y nos pueda llevar con rapidez a la respuesta.

Para realizar una búsqueda web de la manera tradicional, el usuario dirige su navegador al URL de un sitio web de búsqueda. Los principales sitios de búsqueda son: Google, Yahoo! y Bing. A continuación el usuario envía los términos de búsqueda mediante el uso de un formulario. Esta acción provoca que el motor de búsqueda realice una consulta en su base de datos en cuanto a páginas o imágenes relevantes, o cualquier tipo de recurso que se esté buscando, y que devuelva el resultado como una página dinámica. Así, el usuario puede seguir vínculos a las páginas que se hayan encontrado.

La búsqueda web es un interesante tema de discusión, ya que existen implicaciones en cuanto al diseño y el uso de las redes. En primer lugar, está la pregunta de cómo la búsqueda web encuentra las páginas. El motor de búsqueda web debe tener una base de datos de páginas para ejecutar una consulta. Cada página de HTML puede contener vínculos a otras páginas, y todo lo interesante (o por lo menos que se pueda buscar) está vinculado en alguna otra parte. Esto significa que en teoría es posible iniciar con un puñado de páginas y encontrar a todas las demás páginas en la web mediante un recorrido de todas las páginas y vínculos. A este proceso se le conoce como **Rastreo Web** (*Web Crawling***)**. Todos los motores de búsqueda web usan rastreadorse web.

Un problema con el rastreo web es el tipo de páginas que puede encontrar. Es fácil obtener documentos estáticos y seguir los vínculos. Sin embargo, muchas páginas web contienen programas que muestran distintas páginas, dependiendo de la interacción del usuario. Como ejemplo está el catálogo en línea de una tienda. Este catálogo puede contener páginas dinámicas creadas a partir de una base de datos y consultas sobre distintos productos. Este tipo de contenido es diferente a las páginas estáticas que son

fáciles de recorrer. ¿Cómo encuentran los rastreadores web estas páginas dinámicas? La respuesta es que, la mayoría, no lo hacen. A este tipo de contenido oculto se le conoce como **web profunda**. La forma de buscar en la web profunda es un problema abierto que los investigadores están tratando de resolver. Por ejemplo, consulte a Madhavan y colaboradores (2008). Existen también convenciones mediante las cuales los sitios crean una página (conocida como *robots.txt*) para indicar a los rastreadores qué partes de los sitios deben o no visitar.

Una segunda consideración es cómo procesar todos los datos examinados. Para permitir que se ejecuten algoritmos de indexado sobre la masa de datos, hay que almacenar las páginas. Las estimaciones varían, pero se piensa que los principales motores de búsqueda tienen un índice de decenas de miles de millones de páginas que se toman de la parte visible de la web. El tamaño de página promedio se estima en 320 KB. Estas cifran estiman que una copia rastreada de la web tarda cerca de 20 petabytes o 2×10^{16} bytes en almacenarse. Aunque éste es un número enorme, es también una cantidad de datos que se puede almacenar y procesar sin problemas en los centros de datos de Internet (Chang y colaboradores, 2006). Por ejemplo, si el almacenamiento en disco cuesta $20/TB, entonces 2×10^4 TB cuestan $400 000, lo cual no es en sí una cantidad grande para empresas del tamaño de Google, Microsoft y Yahoo! Y mientras se expande la web, los costos de los discos disminuyen en forma drástica, por lo que almacenar toda la web completa puede seguir siendo algo viable para las empresas grandes en el futuro inmediato.

Hacer uso de estos datos es otro asunto. Aquí se puede apreciar cómo puede el XML ayudar a los programas a extraer la estructura de los datos con facilidad, mientras que los formatos específicos conducirán a muchas conjeturas. Existe también el problema de la conversión entre los formatos, e incluso de la traducción entre los lenguajes. Pero incluso conocer la estructura de los datos es sólo parte del problema. Lo difícil es entender su significado. Aquí es donde se puede descubrir mucho valor, empezando con las páginas de resultados más relevantes para las consultas de búsqueda. El objetivo más importante es responder a las preguntas; por ejemplo, dónde comprar un tostador económico pero decente en su ciudad.

Un tercer aspecto de la búsqueda web es que ha llegado a proveer un nivel más alto en cuanto a los nombres. No hay necesidad de recordar un URL largo si es igual de confiable (o tal vez más) buscar una página web con base en el nombre de la persona, suponiendo que seamos mejor al recordar nombres en vez de los URL. Esta estrategia está teniendo cada vez más éxito. De la misma forma que los nombres DNS relegan direcciones IP a las computadoras, la búsqueda web relega direcciones URL a las computadoras. Otra cosa a favor de la búsqueda es que corrige los errores de ortografía y de escritura, mientras que si escribimos un URL mal, recibimos la página incorrecta.

Por último, la búsqueda web nos muestra algo que no tiene mucho que ver con el diseño de las redes, sino con el crecimiento de ciertos servicios de Internet: hay mucho dinero en la publicidad. Éste es el motor económico que ha impulsado el crecimiento de la búsqueda web. El principal cambio de la publicidad impresa es la habilidad de dirigir los anuncios, dependiendo de lo que las personas estén buscando, para incrementar la relevancia de los mismos. Se utilizan variaciones de un mecanismo de subastas para asociar la consulta de búsqueda con el anuncio más valioso (Edelman y colaboradores, 2007). Desde luego que este nuevo modelo ha dado pie a nuevos problemas, como el **fraude del clic**, en el que los programas imitan a los usuarios y hacen clics en anuncios para provocar pagos que no se han obtenido en forma justa.

7.4 AUDIO Y VIDEO DE FLUJO CONTINUO

Las aplicaciones web y la web móvil no son los únicos desarrollos emocionantes en el uso de las redes. Para muchas personas, el audio y el video son el Santo Grial de las redes. Cuando se menciona la palabra “multimedia”, tanto los expertos como los ejecutivos empiezan a emocionarse al mismo tiempo. Los

primeros ven enormes retos técnicos para proveer voz sobre IP y video bajo demanda a todas las computadoras. Los segundos ven ganancias igual de inmensas en ello.

Aunque la idea de enviar audio y video sobre Internet ha estado circulando desde la década de 1970 por lo menos, no fue sino hasta cerca del año 2000 que empezó a crecer de manera desmesurada el tráfico de **audio en tiempo real** y **video en tiempo real**. El tráfico en tiempo real es distinto del tráfico web en cuanto a que se debe reproducir a cierta velocidad predeterminada para ser de utilidad. Después de todo, ver un video en cámara lenta con vaivenes no es la definición de diversión para la mayoría de las personas. Por el contario, la web puede tener interrupciones cortas y el proceso de cargar las páginas puede tardar tiempo, dentro de ciertos límites, sin que ello provoque un problema grave.

Ocurrieron dos cosas para permitir este crecimiento. Primero, las computadoras se volvieron mucho más poderosas y están equipadas con micrófonos y cámaras, lo cual les permite introducir, procesar y producir datos de audio y video con facilidad. Segundo, el ancho de banda disponible para Internet aumentó exponencialmente. Los enlaces de largo recorrido en el núcleo de Internet operan a muchos gigabits/seg, y las tecnologías de ancho de banda y 802.11 inalámbrica llegan a los usuarios al otro extremo de Internet. Estos desarrollos permiten a los ISP transportar enormes niveles de tráfico a través de sus redes troncales, lo cual significa que los usuarios ordinarios se pueden conectar a Internet de 100 a 1000 veces más rápido que con el módem telefónico de 56 kbps.

El aumento desmesurado del ancho de banda provocó que aumentara el tráfico de audio y video, pero por distintas razones. Las llamadas telefónicas ocupan un ancho de banda relativamente pequeño (en principio son 64 kbps, pero es menos cuando se comprime), y a pesar de ello el servicio telefónico ha sido, por tradición, costoso. Las compañías vieron la oportunidad de transportar el tráfico de voz sobre Internet mediante el uso del ancho de banda existente para recortar sus tarifas telefónicas. Las empresas recién iniciadas como Skype vieron la forma de permitir que sus clientes realizaran llamadas telefónicas gratuitas mediante el uso de sus conexiones a Internet. Las compañías telefónicas que recién empezaban vieron una manera económica de transportar las llamadas de voz tradicionales mediante el uso de equipo de redes IP. El resultado fue una explosión en los datos de voz que se transportan a través de redes de Internet, a lo cual se le conoce como **telefonía de Internet** o **voz sobre IP**.

A diferencia del audio, el video ocupa una gran cantidad de ancho de banda. El video en Internet de calidad razonable se codifica mediante compresión a tasas de alrededor de 1 Mbps, y una típica película en DVD contiene 2 GB de datos. Antes del acceso a Internet de banda ancha, enviar películas sobre la red era algo prohibitivo. Pero ya no es así. Con el esparcimiento de la banda ancha, por primera vez los usuarios podían ver video decente en flujo continuo en su hogar. A las personas les encanta hacer esto. Se estima que, en un día común, cerca de una cuarta parte de los usuarios de Internet visitan YouTube, el popular sitio para compartir videos. El negocio de rentas de video ha cambiado a las descargas en línea. Y el tamaño en sí de los videos ha cambiado la composición general del tráfico en Internet. La mayoría de este tráfico viene siendo ya video; se estima que 90% del tráfico en Internet será video dentro de unos cuantos años (Cisco, 2010).

Dado que hay suficiente ancho de banda para transportar audio y video, la cuestión clave para diseñar aplicaciones de flujo continuo y de conferencias es el retardo en la red. El audio y el video necesitan una presentación en tiempo real, lo cual significa que se deben reproducir a una tasa de transferencia predeterminada para ser de utilidad. Los retardos extensos implican que las llamadas que deberían ser interactivas ya no lo serán. El problema está claro si alguna vez usted ha hablado por un teléfono satelital, en donde el retardo de hasta medio segundo es bastante molesto. Para reproducir música y películas a través de la red el retardo absoluto no importa, ya que sólo afecta cuando los medios se empiezan a reproducir. Pero la variación en el retardo, conocida en inglés como **jitter**, sí es importante. El reproductor debe enmascararla, o de lo contrario el audio será incomprensible y el video se verá entrecortado.

En esta sección analizaremos algunas estrategias para manejar el problema del retardo, así como los protocolos para establecer sesiones de audio y video. Después de una introducción al audio y

video digital, dividiremos nuestra presentación en tres casos para los que se utilizan distintos diseños. El primer caso (el más sencillo) es la transmisión en flujo continuo de medios almacenados, como ver un video en YouTube. El siguiente caso en términos de dificultad es la transmisión en flujo continuo de medios en vivo. Dos ejemplos son la radio en Internet y la IPTV, en donde las estaciones de radio y televisión difunden contenido a muchos usuarios en vivo a través de Internet. El último caso (y el más difícil) es el de hacer una llamada como en Skype, o dicho en forma más general, una conferencia interactiva de audio y video.

Por otro lado, el término **multimedia** se utiliza con frecuencia en el contexto de Internet para indicar audio y video. Literalmente, multimedia significa dos o más medios. Esa definición hace de este libro una presentación multimedia, ya que contiene texto y gráficos (las figuras). Sin embargo, quizá eso no es lo que el lector tenía en mente, así que utilizaremos el término "multimedia" para implicar dos o más **medios continuos**; es decir, medios que se tienen que reproducir durante cierto intervalo bien definido. Por lo general los dos medios son video con audio; es decir, imágenes en movimiento con sonido. Muchas personas también hacen referencia al audio puro (como la telefonía o el radio por Internet) como multimedia, aunque es evidente que no es así. En realidad, un mejor término para todos estos casos es el de **medios de flujo continuo**. Sin embargo, seguiremos a la mayoría y consideraremos al audio en tiempo real también como multimedia.

7.4.1 Audio digital

Una onda de audio (sonido) es una onda acústica unidimensional (de presión). Cuando una onda acústica entra al oído, el tambor vibra y hace que los diminutos huesos del oído interno vibren junto con él para enviar pulsos nerviosos al cerebro. El interlocutor percibe estos pulsos como sonido. De una manera similar, cuando una onda acústica golpea a un micrófono, éste genera una señal eléctrica que representa la amplitud del sonido como una función del tiempo.

El rango de frecuencia del oído humano es de 20 Hz a 20 000 Hz. Algunos animales, en particular los perros, pueden escuchar frecuencias más altas. El oído escucha el ruido en forma logarítmica, por lo que la proporción de dos sonidos con potencia A y B se expresa de manera convencional en **dB** (**decibeles**) como la cantidad $10 \log_{10} (A/B)$. Si definimos el límite inferior de audibilidad (una presión de sonido de alrededor de 20 μpascales) para una onda senoidal de 1 kHz como 0 dB, una conversación ordinaria está cerca de los 50 dB y el umbral de dolor es de cerca de 120 dB. El rango dinámico es un factor de más de 1 millón.

El oído es sorprendentemente sensible a las variaciones de sonido que duran sólo unos cuantos milisegundos. En contraste, el ojo no detecta los cambios en el nivel de luz que duran unos cuantos milisegundos. El resultado de esta observación es que la variación del retardo o *jitter* de sólo unos cuantos milisegundos durante la reproducción de contenido multimedia afecta a la calidad del sonido percibido mucho más que a la calidad de la imagen percibida.

El audio digital es una representación digital de una onda de audio que se puede usar para recrearla. Las ondas de audio se pueden convertir a formato digital mediante un **ADC** (**Convertidor Analógico-Digital**, del inglés *Analog-to-Digital Converter*). Éste recibe un voltaje eléctrico como entrada y genera un número binario como salida. En la figura 7-42(a) podemos ver un ejemplo de una onda senoidal. Para representar esta señal en forma digital, podemos tomar muestras de ella cada ΔT segundos, como muestran las alturas de las barras en la figura 7-42(b). Si una onda de sonido no es una onda senoidal pura, sino una superposición lineal de ondas senoidales en donde el componente de frecuencia más alto presente está en f, el teorema de Nyquist (vea el capítulo 2) establece que es suficiente tomar muestras a una frecuencia $2f$. Un muestreo con más frecuencia no tiene valor, ya que las frecuencias más altas que dicho muestreo podría detectar no están presentes.

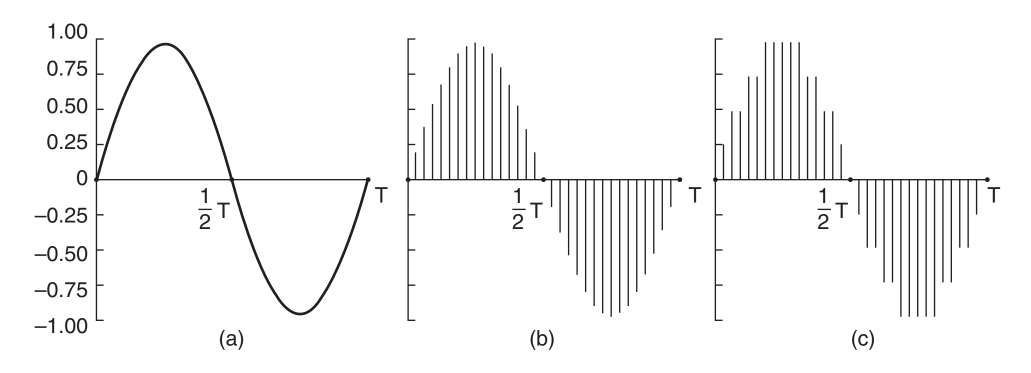


Figura 7-42. (a) Una onda senoidal. (b) Muestreo de la onda senoidal. (c) Cuantización de las muestras a 4 bits.

El proceso inverso toma valores digitales y produce un voltaje eléctrico analógico. Esto se hace mediante un **DAC** (**Convertidor Digital-Analógico**, del inglés *Digital-to-Analog Converter*). Después una bocina puede convertir el voltaje analógico en ondas acústicas, de modo que las personas puedan escuchar sonidos.

Las muestras digitales nunca son exactas. Las de la figura 7-42(c) permiten sólo nueve valores, de -1.00 a $+1.00$ en intervalos de 0.25. Una muestra de 8 bits permitiría 256 valores distintos. Una de 16 bits permitiría 65 536 valores distintos. El error introducido por el número finito de bits por muestra se conoce como **ruido de cuantización**. Si es demasiado grande, el oído lo detecta.

Dos ejemplos bien conocidos en donde se utiliza el sonido muestreado son el teléfono y los discos compactos de audio. La modulación de código de pulso, como la que se usa en el sistema telefónico, emplea muestras de 8 bits que se toman 8 000 veces por segundo. La escala es no lineal para minimizar la distorsión percibida, y con sólo 8 000 muestras/seg se pierden las frecuencias superiores a los 4 kHz. En Norteamérica y Japón se utiliza la codificación de **Ley μ** (*μ-law*). En Europa y a nivel internacional, se utiliza la codificación de **Ley A**. Cada codificación produce una tasa de transferencia de datos de 64 000 bps.

Los CD de audio son digitales con una tasa de muestreo de 44 100 muestras/seg, lo suficiente como capturar frecuencias de hasta 22 050 Hz, una cantidad bastante buena para las personas pero mala para los caninos amantes de la música. Las muestras son de 16 bits cada una y son lineales dentro del rango de amplitudes. Cabe mencionar que las muestras de 16 bits permiten sólo 65 535 valores distintos, incluso cuando el rango dinámico del oído es de más de 1 millón. Por ende, aunque el audio con calidad de CD es mucho mejor que el audio con calidad telefónica, al usar sólo 16 bits por muestra se introduce un ruido de cuantización considerable (aunque no se cubre el rango dinámico completo, se supone que los CD no deben lastimar). Algunos audiófilos fanáticos siguen prefiriendo los discos LP de 33 RPM a los CD, ya que los discos no tienen un corte de frecuencia de Nyquist a los 22 kHz y tampoco tienen ruido de cuantización (pero se pueden rayar con facilidad, a menos que se traten con mucho cuidado). Con 44 100 muestras/seg de 16 bits cada una, el audio con calidad de CD sin comprimir necesita un ancho de banda de 705.6 kbps para señal monoaural y de 1.411 Mbps para señal estéreo.

Compresión de audio

Es común comprimir el audio para reducir las necesidades de ancho de banda y los tiempos de transferencia, aun cuando las tasas de datos de audio son mucho más bajas que las de video. Todos los sistemas de compresión requieren dos algoritmos: uno para comprimir los datos en el origen y otro para descomprimirlos en el destino. En la literatura, estos algoritmos se conocen como los algoritmos de **codificación** y **decodificación**, respectivamente. Nosotros también usaremos esta terminología.

Los algoritmos de compresión exhiben ciertas asimetrías importantes que debemos comprender. Aun cuando estamos considerando primero el audio, estas asimetrías también son válidas para el video. En muchas aplicaciones, un documento multimedia sólo se codificará una vez (cuando se almacena en el servidor multimedia) pero se decodificará miles de veces (cuando los clientes lo reproduzcan). Esta asimetría significa que es aceptable que el algoritmo de codificación sea lento y requiera hardware costoso, siempre y cuando el algoritmo de decodificación sea rápido y no requiera hardware costoso. El operador de un servidor popular de audio (o video) podría estar muy dispuesto a comprar un clúster de computadoras para codificar toda su biblioteca completa, pero obligar a los clientes a que hagan lo mismo para escuchar música o ver películas es algo que quzá no tendrá mucho éxito. Muchos sistemas de compresión prácticos hacen un enorme esfuerzo por lograr que la decodificación sea rápida y simple, incluso al precio de hacer la codificación un proceso lento y complicado.

Por otra parte, en el audio y video en vivo, como las llamadas de voz sobre IP, una codificación lenta es inaceptable. La codificación se debe realizar al instante, en tiempo real. En consecuencia, la multimedia en tiempo real usa distintos algoritmos o parámetros que el audio o los videos almacenados en el disco, a menudo con una compresión mucho menor.

Una segunda asimetría es que el proceso de codificación/decodificación no necesita invertirse. Es decir, al comprimir un archivo de datos, transmitirlo y después descomprimirlo, el usuario espera obtener el original de vuelta, con una precisión de hasta el último bit. En multimedia este requerimiento no existe. Por lo general es aceptable que la señal de audio (o video) después del proceso de codificación y decodificación sea ligeramente distinta a la original, siempre y cuando suene (o se vea) igual. Cuando la salida decodificada no es exactamente igual a la entrada original, se dice que es un sistema **con pérdida** (*lossy*). Si la entrada y la salida son idénticas, se dice que el sistema es **sin pérdida** (*lossless*). Los sistemas con pérdida son importantes, ya que aceptar una pequeña cantidad de pérdida de información por lo general significa una enorme ganancia en términos de la razón de compresión posible.

Históricamente, el ancho de banda de largo recorrido en la red telefónica era muy costoso, por lo que hay una muestra de trabajo considerable respecto a los **vocoders** (abreviación de “codificadores de voz”), los cuales comprimen audio para el caso especial de la voz. La voz humana tiende a estar en el rango de 600 Hz a 6 000 Hz y se produce mediante un proceso mecánico que depende del tracto vocal, la lengua y la mandíbula del orador. Algunos vocoders utilizan modelos del sistema vocal para reducir la voz a unos cuantos parámetros (por ejemplo, los tamaños y formas de varias cavidades) y a una tasa de datos pequeña de hasta 2.4 kbps. Sin embargo, la forma en que trabajan estos vocoders queda fuera del alcance de este libro.

Nos concentraremos en el audio que se envía a través de Internet, que por lo general tiene una calidad cercana a la del CD. También es conveniente reducir las tasas de datos para este tipo de audio. A 1.411 Mbps, el audio estéreo ocuparía muchos enlaces de banda ancha y dejaría menos espacio para el video y otros tipos de tráfico web. Su tasa de datos con compresión se puede reducir por un orden de magnitud, con una pérdida de calidad que puede variar desde muy poca hasta imperceptible.

La compresión y la descompresión requieren el procesamiento de señales. Por fortuna, las computadoras pueden procesar con facilidad el sonido y las películas digitalizadas mediante software. De hecho, existen docenas de programas que permiten a los usuarios desplegar, editar, mezclar y almacenar medios de varias fuentes. Esto ha provocado que haya grandes cantidades de música y películas disponibles en Internet (no todo es legal), y se han originado numerosas demandas legales por parte de los artistas y dueños de los derechos reservados.

Se han desarrollado muchos algoritmos de compresión de audio. Tal vez los formatos más populares sean **MP3** (**Capa de audio 3 de MPEG**, del inglés *MPEG audio layer 3*) y **AAC** (**Codificación de Audio Avanzada**, del inglés *Advanced Audio Coding*), que se transportan en archivos **MP4** (**MPEG-4**). Para evitar confusiones, cabe mencionar que MPEG provee compresión de audio y video. MP3 se refiere a la parte correspondiente a la compresión de audio (parte 3) del estándar MPEG-1, no a la tercera versión

de MPEG. De hecho, nunca se liberó una tercera versión de MPEG, sólo MPEG-1, MPEG-2 y MPEG-4. AAC es la sucesora de MP3 y la codificación de audio predeterminada que se utiliza en MPEG-4. MPEG-2 permite audio tanto MP3 como AAC. ¿Está claro ahora? Lo bueno sobre los estándares es que hay muchos de dónde elegir. Y si no le gusta ninguno de ellos, sólo tiene que esperar uno o dos años.

La compresión de audio se puede realizar de dos formas. En la **codificación de forma de onda** la señal se transforma de manera matemática en sus componentes de frecuencia mediante una transformación de Fourier. En la figura 2-1(a) del capítulo 2 se muestra una función de ejemplo de tiempo y sus amplitudes de Fourier. Por lo tanto, la amplitud de cada componente se codifica en una forma mínima. El objetivo es reproducir la forma de onda de manera precisa en el otro extremo, utilizando la menor cantidad posible de bits.

La otra forma, **codificación perceptual**, aprovecha ciertas fallas del sistema auditivo humano para codificar una señal con el fin de que suene de la misma forma para un oyente humano, aunque dicha señal luzca de manera muy distinta en un osciloscopio. La codificación perceptual se basa en la ciencia de **psicoacústica** —cómo perciben las personas el sonido. Tanto MP3 como AAC se basan en la codificación perceptual.

La propiedad clave de la codificación perceptual es que algunos sonidos pueden **enmascarar** otros sonidos. Imagine que está difundiendo un concierto de flauta en vivo en un día caluroso de verano. De repente, un grupo de trabajadores que está cerca enciende sus martillos perforadores y comienza a romper la calle. Ya nadie puede escuchar la flauta. Sus sonidos han sido enmascarados por los de los martillos perforadores. Para fines de transmisión, ahora es suficiente con codificar sólo la banda de frecuencia utilizada por los martillos perforadores, pues de cualquier forma las personas no pueden escuchar la flauta. A esto se le conoce como **enmascaramiento de frecuencia** —la capacidad que tiene un sonido fuerte en una banda de frecuencia de ocultar un sonido más suave en otra banda de frecuencia, el cual podría ser audible si el sonido fuerte no estuviera presente. De hecho, incluso después de que los martillos perforadores pararan, la flauta no se escucharía por un periodo corto debido a que el oído reduce su ganancia cuando los martillos comienzan y toma un tiempo finito para aumentarlo de nuevo. Este efecto se conoce como **enmascaramiento temporal**.

Para hacer que estos efectos sean más cuantitativos, imagine el experimento 1. Una persona en un salón silencioso se pone unos audífonos que están conectados a la tarjeta de sonido de una computadora. Ésta genera una onda senoidal pura a 100 Hz con una potencia baja, pero incrementa la potencia de manera gradual. Se le indica a la persona que pulse una tecla cuando escuche el tono. La computadora graba el nivel de potencia actual y después repite el experimento a 200 Hz, 300 Hz y demás frecuencias hasta el límite del oído humano. Después de sacar un promedio a partir de repetir el experimento con varias personas, se obtiene un gráfico log-log de cuánta potencia se necesita para que un tono sea audible por los humanos, el cual luce como se muestra en la figura 7-43(a). Una consecuencia directa de esta curva es que nunca es necesario codificar ninguna frecuencia cuya potencia esté por debajo del umbral de audibilidad. Por ejemplo, si la potencia de 100 Hz fuera de 20 dB en la figura 7-43(a), se podría omitir de la salida sin ninguna pérdida perceptible de calidad debido a que 20 dB a 100 Hz están debajo del nivel audible.

Ahora considere el experimento 2. La computadora realiza otra vez el experimento 1, pero esta vez con una onda senoidal de amplitud constante a, digamos, 150 Hz, sobrepuesta en la frecuencia de prueba. Lo que descubrimos es que se eleva el umbral de audibilidad para las frecuencias que están cerca de 150 Hz, como se muestra en la figura 7-43(b).

La consecuencia de esta nueva observación es que, al mantener un registro de cuáles señales están siendo enmascaradas por señales más poderosas en bandas de frecuencia cercanas, podemos omitir cada vez más frecuencias en la señal codificada, con lo cual ahorramos bits. En la figura 7-43, la señal a 125 Hz se puede omitir por completo de la salida y nadie notará la diferencia. Aunque una señal poderosa se detenga en alguna banda de frecuencia, si conocemos sus propiedades de enmascaramiento temporal, podemos continuar omitiendo las frecuencias enmascaradas por algún intervalo mientras el oído se recu-

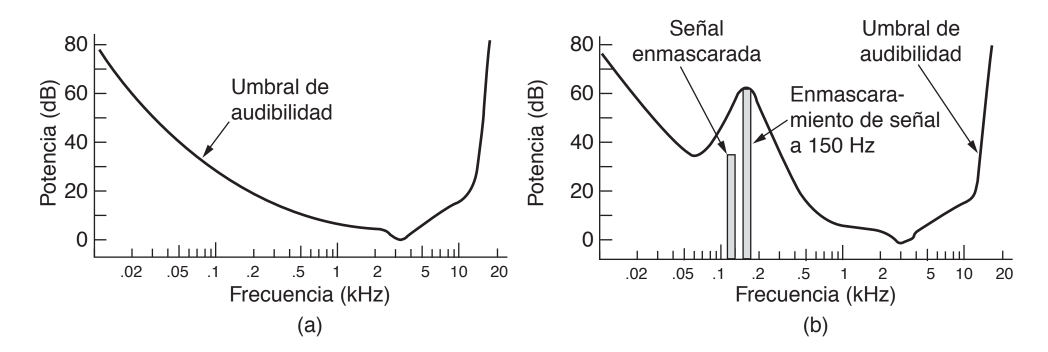


Figura 7-43. (a) El umbral de audibilidad como función de la frecuencia. (b) El efecto de enmascaramiento.

pera. La esencia de MP3 y de AAC es transformar mediante un análisis de Fourier el sonido para obtener la potencia de cada frecuencia y después transmitir sólo las frecuencias no enmascaradas, codificando éstas con la menor cantidad posible de bits.

Con esta información como base, ahora podemos ver cómo se realiza la codificación. La compresión de audio se realiza mediante el muestreo de la forma de onda a una tasa desde 8 hasta 96 kHz para AAC, a menudo a 44.1 kHz, para imitar el sonido de CD. El muestreo se puede realizar en uno (mono) o en dos (estéreo) canales. Enseguida se selecciona la tasa de salida de bits. MP3 puede comprimir un CD de *rock and roll* estéreo hasta 96 kbps con poca pérdida perceptible en la calidad, incluso para los fans de *rock and roll* sin deficiencias auditivas. Para un concierto de piano, se requiere AAC con al menos 128 kbps. La diferencia se debe a que la relación señal —ruido para el *rock and roll* es mucho más alta que la de un concierto de piano (en el sentido de la ingeniería). También es posible elegir tasas de salida más bajas y aceptar cierta pérdida en la calidad.

Las muestras se procesan en lotes pequeños. Cada lote se pasa a través de un banco de filtros digitales para obtener bandas de frecuencia. La información de frecuencia se introduce en un modelo psicoacústico para determinar las frecuencias enmascaradas. Después, los bits disponibles se dividen entre las bandas; la mayoría de los bits se asignan a las bandas con la mayor potencia espectral no enmascarada, a las bandas no enmascaradas con menos potencia espectral se les asignan menos bits y a las bandas enmascaradas no se les asignan bits. Por último, los bits se codifican mediante la codificación de Huffman, que asigna códigos cortos a números que aparecen con frecuencia, y códigos largos a aquellos que ocurren con poca frecuencia. Hay todavía muchos más detalles para el lector curioso. Para obtener más información, consulte a Brandenburg (1999).

7.4.2 Video digital

Ahora que sabemos todo sobre el oído, es tiempo de pasar al ojo. El ojo humano tiene la propiedad de que cuando aparece una imagen en la retina, ésta se retiene unos segundos antes de decaer. Si se dibuja una secuencia de imágenes a razón de 50 imágenes/seg, el ojo no detecta que está viendo imágenes discretas. Todos los sistemas de video explotan este principio para producir imágenes en movimiento.

La representación digital más simple del video es una secuencia de tramas, cada uno de las cuales consiste en una rejilla rectangular de elementos de imagen, o **píxeles**. Cada píxel puede ser de un solo bit, para representar ya sea negro o blanco. Sin embargo, la calidad de dicho sistema es pésima. Pruebe a usar su editor de imágenes favorito y convierta los píxeles de una imagen a colores en blanco y negro (pero *no* en tonos de gris).

El siguiente paso es utilizar 8 bits por píxel para representar 256 niveles de gris. Este esquema produce video en "blanco y negro" de alta calidad. Para el video a colores, muchos sistemas usan 8 bits para cada uno de los componentes primarios de color: rojo, verde y azul (RGB). Esta representación es posible debido a que cualquier color se puede construir a partir de una superposición lineal de rojo, verde y azul con las intensidades apropiadas. Con 24 bits por píxel, hay cerca de 16 millones de colores, que es más de lo que el ojo humano puede distinguir.

En los monitores de computadora y televisiones LCD de color, cada píxel discreto está compuesto por un subpíxel rojo, uno verde y uno azul, con muy poco espacio entre ellos. Para visualizar las tramas se ajusta la intensidad de los subpíxeles y el ojo mezcla los componentes de color.

Las tasas de tramas comunes son: 24 tramas/seg (heredada de los filmes cinematográficos de 35 mm), 30 tramas/seg (heredada de las televisiones NTSC en Estados Unidos.) y 30 tramas/seg (heredada del sistema de televisión PAL utilizado en casi todo el resto del mundo). (Para los verdaderos quisquillosos, la televisión a color NTSC opera a 29.97 tramas/seg. El sistema en blanco y negro original operaba a 30 tramas/seg, pero cuando se introdujo el color, los ingenieros necesitaban un poco de ancho de banda adicional para la señalización, por lo que redujeron la tasa de tramas a 29.97. Los videos NTSC destinados para las computadoras en realidad usan 30). PAL se inventó después de NTSC y en realidad usa 25 000 tramas/seg. Para completar esta historia, hay un tercer sistema llamado SECAM que se utiliza en Francia, las regiones en África de habla francesa y Europa Oriental. Se introdujo por primera vez en Europa Oriental gracias a la entonces Alemania Oriental Comunista, de modo que las personas de Alemania Oriental no pudieran ver la televisión de Alemania Occidental (PAL) para que no tuvieran mala influencia. Pero muchos de estos países están cambiando a PAL. La tecnología y la política a su máxima expresión.

En realidad, para la televisión de difusión no son suficientes 25 tramas/seg para un movimiento continuo, por lo que las imágenes se dividen en dos **campos**, uno con las líneas de exploración de numeración impar y otro con las de numeración par. Los dos campos (media resolución) se difunden en forma secuencial, con lo cual se obtienen casi 60 (NTSC) o exactamente 50 (PAL) campos/seg, un sistema conocido como **entrelazado**. Los videos destinados a verse en una computadora son **progresivos**; es decir, no usan el entrelazado debido a que los monitores de computadora tienen búferes en sus tarjetas de gráficos, de modo que el CPU puede colocar una nueva imagen en el búfer 30 veces/seg, al tiempo que la tarjeta de gráficos vuelve a dibujar la pantalla 50 o incluso 100 veces/seg para eliminar el parpadeo. Las televisiones analógicas no tienen un búfer de tramas como las computadoras. Cuando se muestra un video entrelazado con movimiento rápido en una computadora, se podrán ver líneas horizontales cortas cerca de los bordes pronunciados; a este efecto se le conoce como **efecto peine** (*combing*).

Los tamaños de trama utilizados para el video que se envía a través de Internet varían mucho por la sencilla razón de que las tramas más grandes requieren más ancho de banda, que no siempre puede estar disponible. El video de baja resolución podría ser de 320 por 240 píxeles, y el video de "pantalla completa" es de 640 por 480 píxeles. Estas medidas se aproximan a las de los primeros monitores de computadora y la televisión NTSC, respectivamente. La **relación de aspecto**, o relación de anchura a altura, de 4:3 es igual a la de una televisión estándar. Se pueden descargar videos **HDTV** (**Televisión de Alta Definición**, del inglés *High-Definition TeleVision*) con 1280 por 720 píxeles. Estas imágenes de "pantalla panorámica" (*widescreen*) tienen una relación de aspecto de 16:9 y se asemejan más a la relación de aspecto 3:2 del cine. Como comparación, el video de un DVD estándar es por lo general de 720 por 480 píxeles, y el video en los discos Blu-ray es por lo general HDTV a 1080 por 720 píxeles.

En Internet, el número de píxeles es sólo parte de la historia, puesto que los reproductores de medios pueden presentar la misma imagen a distintos tamaños. El video es sólo otra ventana en la pantalla de una computadora que se puede maximizar o reducir. La función que desempeñan los píxeles

adicionales es incrementar la calidad de la imagen, de modo que no se vea borrosa al expandirse. Sin embargo, muchos monitores pueden mostrar imágenes (y también videos) con muchos más pixeles que inclusive la HDTV.

Compresión de video

Con base en nuestro análisis del video digital, debe ser obvio que la compresión es imprescindible para enviar video a través de Internet. Incluso un video de calidad estándar con tramas de 640 por 480 píxeles, 24 bits de información de color por píxel y 30 tramas/seg, ocupa más de 200 Mbps. Esto excede por mucho el ancho de banda a través del cual se conectan a Internet la mayoría de las empresas, sin mencionar los usuarios domésticos, y esto para un solo flujo de video. Como la transmisión de video sin comprimir está totalmente fuera de consideración, por lo menos a través de redes de área amplia, la única esperanza es que se pueda usar la compresión masiva. Por fortuna, un gran conjunto de investigaciones durante las últimas décadas han conducido al desarrollo de muchas técnicas y algoritmos de compresión que permiten transmitir video.

Se usan muchos formatos para el video que se envía a través de Internet; algunos son propietarios y otros son estándar. La codificación más popular es MPEG en sus diversas formas. Es un estándar abierto que se encuentra en los archivos con extensiones mpg y mp4, así como en otros formatos contenedores. En esta sección examinaremos el formato MPEG para estudiar cómo se lleva a cabo la compresión de video. Para empezar, analizaremos la compresión de imágenes fijas con JPEG. Un video es sólo una secuencia de imágenes (más el sonido). Una manera de comprimir video es codificar cada imagen en forma sucesiva. En una primera aproximación, MPEG es sólo la codificación JPEG de cada trama, junto con ciertas características adicionales para eliminar la redundancia entre tramas.

El estándar JPEG

El estándar **JPEG** (**Grupo Unido de Expertos en Fotografía**, del inglés *Joint Photographic Experts Group*) para comprimir imágenes fijas de tono continuo (es decir, fotografías) se desarrolló gracias a los expertos fotográficos que trabajan bajo los auspicios unidos de la ITU, ISO e IEC, otro organismo de estándares. Se utiliza mucho (busque archivos con la extensión jpg) y con frecuencia ofrece relaciones de compresión de 10:1 o mejores para las imágenes naturales.

JPEG se define en el Estándar Internacional 10918. En realidad es más como una lista de compras que un solo algoritmo, pero de los cuatro modos que se definen sólo el modo secuencial con pérdida es relevante para nuestro análisis. Además, nos concentraremos en la forma en que se utiliza JPEG comúnmente para codificar imágenes de video RGB de 24 bits, y omitiremos algunas de las opciones y detalles por cuestión de simpleza.

El algoritmo se ilustra en la figura 7-44. El paso 1 es la preparación del bloque. Con fines de especificación, supongamos que la entrada JPEG es una imagen RGB de 640 $\times$ 480 con 24 bits/píxel, como se muestra en la figura 7-44(a). RGB no es el mejor modelo de color para utilizar en la compresión. El ojo es mucho más sensible a la **luminosidad** o brillo de las señales de video que a la **crominancia** o color de las mismas. Por ende, calculamos primero la luminosidad *Y* y las dos crominancias, *Cb* y *Cr*, a partir de los componentes *R*, *G* y *B*. Las siguientes fórmulas se utilizan para valores de 8 bits que varían de 0 a 255:

$$Y = 165 + 0.26R + 0.50G + 0.09B$$
$$Cb = 128 + 0.15R - 0.29G - 0.44B$$
$$Cr = 128 + 0.44R - 0.37G + 0.07B$$

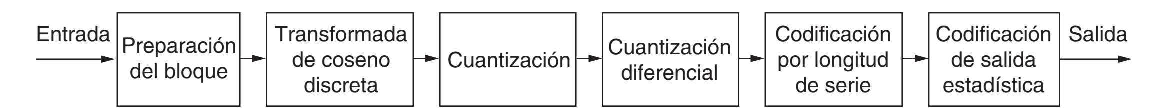


Figura 7-44. Pasos en la codificación secuencial con pérdida de JPEG.

Se construyen matrices separadas para *Y*, *Cb* y *Cr*. A continuación se promedian bloques cuadrados de cuatro píxeles en las matrices *Cb* y *Cr* para reducirlas a 320 × 240. Esta reducción tiene pérdidas, pero el ojo apenas si las nota debido a que responde a la luminosidad más que a la crominancia. Sin embargo, comprime la cantidad total de datos por un factor de dos. Ahora se restan 128 a cada elemento de las tres matrices para colocar un 0 en la parte media del rango. Por último, cada matriz se divide en bloques de 8 × 8. La matriz *Y* tiene 4 800 bloques; las otras dos tienen 1200 bloques cada una, como se muestra en la figura 7-45(b).

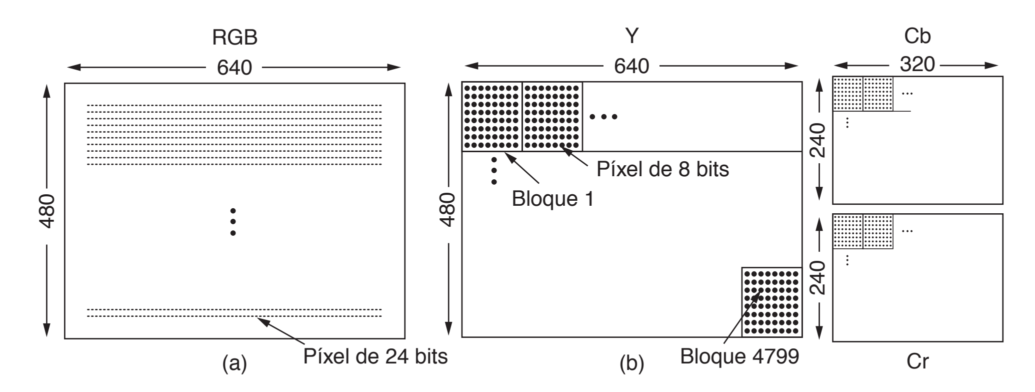


Figura 7-45. (a) Datos de entrada RGB. (b) Después de la preparación del bloque.

El paso 2 de la codificación JPEG es aplicar una **DCT** (**Transformada de Coseno Discreta**, del inglés *Discrete Cosine Transformation*) a cada uno de los 7 200 bloques por separado. La salida de cada DCT es una matriz de 8 × 8 de coeficientes DCT. El elemento DCT (0, 0) es el valor promedio del bloque. Los demás elementos indican cuánta potencia espectral está presente en cada frecuencia espacial. Por lo general estos elementos decaen rápidamente con la distancia desde el origen, (0, 0), como lo sugiere la figura 7-46.

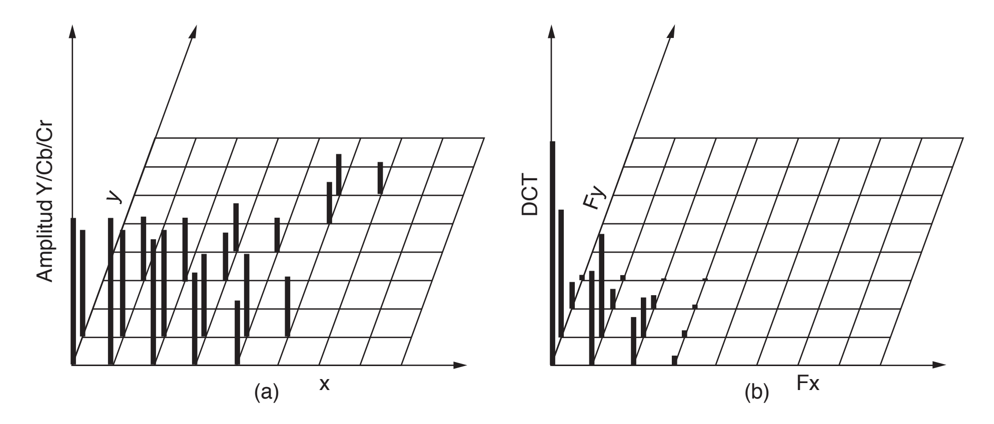
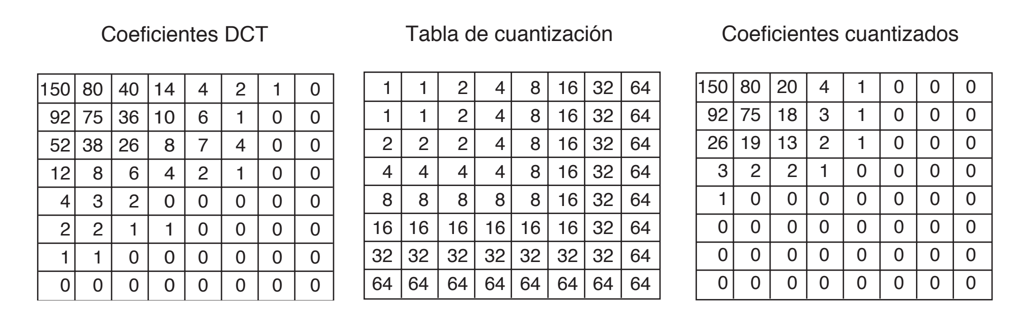


Figura 7-46. (a) Un bloque de la matriz *Y*. (b) Los coeficientes DCT.

Una vez completa la DCT, la codificación JPEG avanza al paso 3, conocido como **cuantización**, en donde se eliminan los coeficientes DCT menos importantes. Esta transformación (con pérdida) se realiza dividiendo cada uno de los coeficientes en la matriz DCT de 8 × 8 entre un peso que se toma de una tabla. Si todos los pesos son 1, la transformación no hace nada. Pero si los pesos aumentan en forma abrupta a partir del origen, las frecuencias espaciales más altas se eliminan rápidamente.

En la figura 7-47 se muestra un ejemplo de este paso. Aquí podemos ver la matriz DCT inicial, la tabla de cuantización y el resultado que se obtiene al dividir cada elemento DCT entre el elemento de la tabla de cuantización correspondiente. Los valores en la tabla de cuantización no son parte del estándar JPEG. Cada aplicación debe proveer sus propios valores, lo cual le permite controlar el grado de pérdida de calidad debido a la compresión.



Coeficientes DCT

150	80	40	14	4	2	1	0
92	75	36	10	6	1	0	0
52	38	26	8	7	4	0	0
12	8	6	4	2	1	0	0
4	3	2	0	0	0	0	0
2	2	1	1	0	0	0	0
1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Tabla de cuantización

1	1	2	4	8	16	32	64
1	1	2	4	8	16	32	64
2	2	2	4	8	16	32	64
4	4	4	4	8	16	32	64
8	8	8	8	8	16	32	64
16	16	16	16	16	16	32	64
32	32	32	32	32	32	32	64
64	64	64	64	64	64	64	64

Coeficientes cuantizados

150	80	20	4	1	0	0	0
92	75	18	3	1	0	0	0
26	19	13	2	1	0	0	0
3	2	2	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Figura 7-47. Cálculo de los coeficientes DCT cuantizados.

El paso 4 reduce el valor (0, 0) de cada bloque (el de la esquina superior izquierda) y lo reemplaza con la cantidad que difiere del elemento correspondiente en el bloque anterior. Como estos elementos son los promedios de sus bloques respectivos, deben cambiar con lentitud, por lo que al tomar los valores diferenciales se deben reducir la mayoría de ellos a valores pequeños. No se calculan diferenciales a partir de los otros valores.

El paso 5 pone en línea los 64 elementos y aplica la codificación por longitud de serie a la lista. Al explorar el bloque de izquierda a derecha y después de arriba hacia abajo no se concentrarán todos los ceros en un solo punto, por lo que se utiliza un patrón de exploración en zig zag, como se muestra en la figura 7-48. En este ejemplo, el patrón de zig zag produce 38 ceros consecutivos al final de la matriz. Esta cadena se puede reducir a un solo conteo que indique que hay 38 ceros, una técnica conocida como **codificación por longitud de serie**.

Ahora tenemos una lista de números que representan la imagen (en espacio de transformada). El paso 6 codifica mediante Huffman los números para su almacenamiento o transmisión, en donde se asignan códigos más cortos a los números comunes que a los no comunes.

JPEG puede parecer complicado, pero esto es porque en realidad *es* complicado. Aún así, los beneficios de una compresión de hasta 20:1 hacen que valga la pena. Para decodificar una imagen JPEG se requiere ejecutar el algoritmo al revés. JPEG es aproximadamente simétrico: la decodificación tarda casi lo mismo que la codificación. Esta propiedad no se aplica a todos los algoritmos de compresión, como veremos a continuación.

El estándar MPEG

Finalmente llegamos al núcleo del asunto: los estándares **MPEG** (**Grupo de Expertos en Imágenes en Movimiento**, del inglés *Motion Picture Experts Group*). Aunque existen muchos algoritmos propieta-

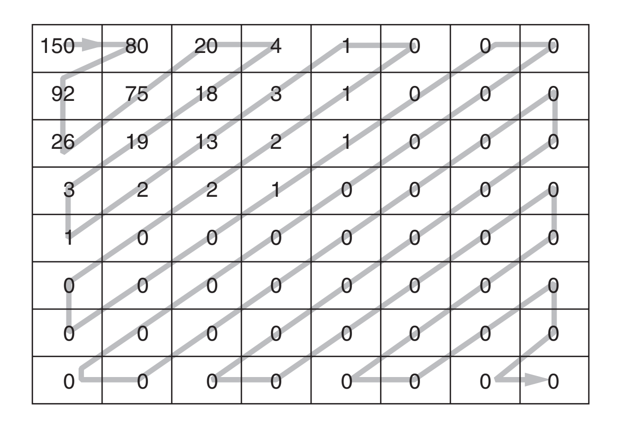


Figura 7-48. El orden en el que se transmiten los valores cuantizados.

rios, estos estándares definen los principales algoritmos que se utilizan para comprimir videos. Han sido estándares internacionales desde 1993. Como las películas contienen tanto imágenes como sonido, MPEG puede comprimir audio y video. Ya examinamos la compresión de audio y la compresión de imágenes fijas, por lo que ahora examinaremos la compresión de video.

El estándar MPEG-1 (que incluye audio MP3) se publicó por primera vez en 1993 y aún es muy utilizado. Su objetivo era producir una salida con calidad de grabadora de video comprimida a una relación de 40:1, a tasas de cerca de 1 Mbps. Este video es adecuado para un uso amplio en Internet en los sitios web. No se preocupe si no recuerda las grabadoras de video; MPEG-1 también se utilizaba para guardar películas en los CD cuando existían. Si no recuerda lo que son los CD, tendremos que pasar a MPEG-2.

El estándar MPEG2, que se liberó en 1996, estaba diseñado para comprimir video de calidad de difusión. Es muy común en la actualidad, ya que se utiliza como la base para el video codificado en los DVD (que inevitablemente encuentra su camino hacia Internet) y para la televisión de difusión digital (como DVB). Por lo general, el video con calidad de DVD se codifica a tasas de 4-8 Mbps.

El estándar MPEG-4 tiene dos formatos de video. El primer liberado en 1999, codifica video con una representación basada en objetos. Esto permite mezclar imágenes naturales y sintéticas junto con otros tipos de medios; por ejemplo, un meteorólogo parado enfrente de un mapa meteorológico. Con esta estructura, es fácil permitir que los programas interactúen con los datos de la película. El segundo formato, que se liberó en 2003, se conoce como **H.264** o **AVC** (**Codificación de Video Avanzada**, del inglés *Advanced Video Coding*). Su objetivo es codificar video a la mitad de la tasa de los primeros codificadores pero con el mismo nivel de calidad, para soportar mejor la transmisión de video a través de las redes. Este codificador se usa para la HDTV en la mayoría de los discos Blu-ray.

Los detalles de todos estos estándares son muchos y variados. Los estándares más recientes también tienen muchas más características y opciones de codificación que los primeros estándares. Sin embargo, no entraremos en detalles sobre esto. En su mayor parte, las ganancias en cuanto a compresión de video a través del tiempo provienen de muchas mejoras pequeñas, en vez de cambios fundamentales en la forma en que se comprime el video, Por lo tanto, bosquejaremos los conceptos en general.

MPEG comprime tanto audio como video. Como los codificadores de audio y de video trabajan de manera independiente, hay un problema a la hora de sincronizar los dos flujos en el receptor. La solución es tener un solo reloj que envíe estampas de tiempo con la hora actual a ambos codificadores. Estas estampas de tiempo se incluyen en la salida codificada y se propagan hasta el receptor, quien puede usarlas para sincronizar los flujos de audio y de video.

La compresión de video MPEG aprovecha dos tipos de redundancias que existen en las películas: espacial y temporal. Para utilizar la redundancia espacial, simplemente se codifica cada trama por separado

con JPEG. Esta metodología se utiliza en ocasiones, en especial cuando se requiere el acceso aleatorio a cada trama, como cuando se editan las producciones de video. En este modo se logran niveles de compresión JPEG.

Podemos obtener una compresión adicional si aprovechamos el hecho de que, con frecuencia, las tramas consecutivas son casi idénticas. Este efecto es menor de lo que podría parecer a primera instancia, ya que muchos directores de películas realizan cortes entre las escenas cada 3 o 4 segundos (tome el tiempo en un fragmento de película y cuente las escenas). Sin embargo, las series de 75 o más tramas muy similares ofrecen el potencial de una importante reducción con sólo codificar cada trama por separado con JPEG.

En las escenas en las que la cámara y el fondo están fijos y sólo se mueven uno o dos actores con lentitud alrededor de la escena, casi todos los píxeles serán idénticos de una trama a otra. Aquí bastaría con sólo restar cada trama a la anterior y aplicar JPEG sobre la diferencia. Sin embargo, en las escenas en donde la cámara hace un barrido o un acercamiento, esta técnica es deficiente. Lo que se requiere es una manera de compensar este movimiento. Esto es precisamente lo que hace MPEG; es la principal diferencia entre MPEG y JPEG.

La salida MPEG consiste en tres tipos de tramas:

1. Tramas I (intracodificadas): imágenes fijas comprimidas y autocontenidas.
2. Tramas P (predictivas): diferencia bloque por bloque con las tramas anteriores.
3. Tramas B (bidireccionales): diferencias bloque por bloque entre las tramas anteriores y futuras.

Las tramas I son sólo imágenes fijas. Se puede codificar con JPEG o algo similar. Es valioso hacer que aparezcan tramas I en el flujo de salida en forma periódica (por ejemplo, una o dos veces por segundo) por tres razones. La primera es que MPEG se puede usar para una transmisión multidifusión, en donde los espectadores ven cuando quieren. Si todas las tramas dependieran de sus predecesores hasta llegar a la primera trama, cualquiera que se perdiera la primera nunca podría decodificar las tramas subsiguientes. En segundo lugar, si se recibiera cualquier trama por error, no sería posible continuar con la decodificación: todo de ahí en adelante sería basura incomprensible. En tercer lugar, sin las tramas I, al realizar un avance rápido o al rebobinar el decodificador tendría que calcular cada trama por la que pasara para conocer el valor completo de la trama en donde se detuviera.

Por el contrario, las tramas P codifican las diferencias entre tramas. Se basan en la idea de los **macrobloques**, que cubren, por ejemplo, 16×16 píxeles en espacio de luminosidad y 8×8 píxeles en espacio de crominancia. Para codificar un macrobloque se busca la trama anterior para éste o algo que sea sólo ligeramente diferente de éste.

En la figura 7-49 se muestra un ejemplo de dónde serían útiles las tramas P. Aquí podemos ver tres tramas consecutivas que tienen el mismo fondo, pero difieren en cuanto a la posición de una persona. Los macrobloques que contienen la escena de fondo coincidirán en forma exacta, pero los macrobloques que contienen a la persona estarán desplazados en posición por cierta cantidad desconocida y habrá que rastrearlos.

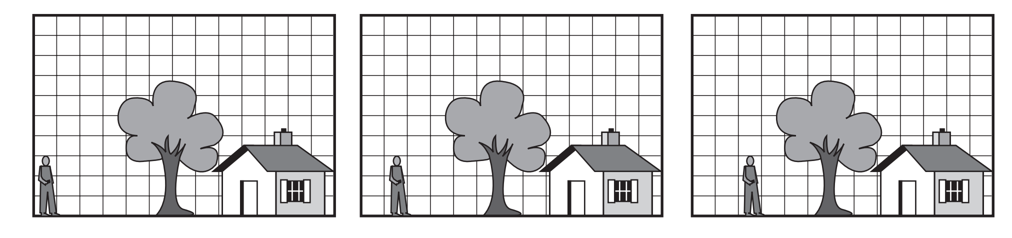


Figura 7-49. Tres tramas consecutivas.

Los estándares MPEG no especifican cómo buscar, qué tan lejos hacerlo o cómo tiene que ser una buena coincidencia para que cuente. Esto depende de cada implementación. Por ejemplo, una implementación podría buscar un macrobloque en la posición actual en la trama anterior, y todas las demás posiciones desplazadas por $\pm\Delta x$ en al dirección x y $\pm\Delta y$ en la dirección y. Para cada posición se podría calcular el número de coincidencias en la matriz de luminosidad. La posición con la puntuación más alta se declararía ganadora, siempre y cuando estuviera por encima de cierto umbral predefinido. En caso contrario, se diría que falta el macrobloque. Desde luego que existen algoritmos mucho más sofisticados.

Si se encuentra un macrobloque, se codifica tomando la diferencia entre su valor actual y el de la trama anterior (tanto para luminosidad como para crominancia). Después, estas matrices de diferencias están sujetas a la transformación de coseno discreta, cuantización, codificación por longitud de serie y codificación de Huffman, como de costumbre. El valor para el macrobloque en el flujo de salida es entonces el vector de movimiento (qué tanto se movió el macrobloque de su posición anterior en cada dirección), seguido de la codificación de su diferencia. Si el macrobloque no se encuentra en la trama anterior, el valor actual se codifica, justo igual que en una trama I.

Es evidente que este algoritmo es muy asimétrico. Una implementación tiene la libertad de probar todas las posiciones plausibles en la trama anterior si lo desea, en un desesperado intento por localizar cada último macrobloque, sin importar a dónde se haya movido. Este método minimizará el flujo MPEG codificado a expensas de una codificación muy lenta. Este método podría estar bien para codificar una sola vez una biblioteca de películas, pero sería terrible para las videoconferencias en tiempo real.

De manera similar, cada implementación tiene la libertad de decidir lo que constituye un macrobloque "encontrado". Esta libertad permite a los implementadores competir por la calidad y velocidad de sus algoritmos, pero siempre producir salida en conformidad con MPEG.

Hasta ahora, la decodificación MPEG es bastante simple. Decodificar tramas I es algo similar a decodificar imágenes JPEG. Para decodificar tramas P, el decodificador tiene que colocar en búfer las tramas anteriores, de modo que pueda crear la nueva en un búfer separado, con base en los macrobloques completamente codificados y los macrobloques que contienen diferencias de las tramas anteriores. La nueva trama se ensambla macrobloque por macrobloque.

Las tramas B son similares a las tramas P, excepto que permiten que el macrobloque de referencia esté en tramas anteriores o posteriores. Esta libertad adicional permite una compensación mejorada del movimiento. Por ejemplo, es útil cuando los objetos pasan en frente de, o detrás de, otros objetos. Para realizar la codificación de trama B, el codificador necesita guardar una secuencia de tramas en memoria a la vez: tramas anteriores, la actual que se va a codificar y las futuras. De manera similar, decodificar es un proceso más complicado y agrega cierto retardo. Esto se debe a que una trama B dada no se puede decodificar sino hasta que se decodifiquen las tramas sucesivas de las que depende. Así, aunque las tramas B ofrecen la mejor compresión, no siempre se utilizan debido a su mayor complejidad y requerimientos de búfer.

Los estándares MPEG contienen muchas mejoras a estas técnicas para lograr excelentes niveles de compresión. Se puede usar AVC para comprimir video en relaciones mayores a 50:1, lo cual reduce los requerimientos de ancho de banda de la red con base en el mismo factor. Para obtener más información sobre AVC, consulte a Sullivan y Wiegand (2005).

7.4.3 Medios almacenados de flujo continuo (*streaming*)

Ahora pasemos a las aplicaciones de red. Nuestro primer caso es el de los medios de flujo continuo que ya están almacenados en archivos. El ejemplo más común de esto es ver videos por Internet. Es una forma de **VoD** (**Video bajo Demanda**, del inglés *Video on Demand*). Otras formas de video bajo demanda usan una red de proveedor que está separada de Internet para ofrecer las películas (por ejemplo, la red de la televisión por cable).

En la siguiente sección analizaremos los medios en vivo de flujo continuo; por ejemplo, IPTV de difusión y radio de Internet. Después analizaremos el tercer caso de las conferencias en tiempo real. Como ejemplo tenemos una llamada de voz sobre IP o una conferencia de video con Skype. Estos tres casos imponen requerimientos cada vez más estrictos en cuanto a la forma en que podemos entregar el audio y el video a través de Internet, ya que debemos poner cada vez más atención al retardo y a la variación del mismo (*jitter*).

La Internet está llena de sitios de música y video que transmiten mediante flujo continuo archivos de medios almacenados. En realidad, la manera más sencilla de manejar medios almacenados es *no* transmitirlos por flujo continuo. Imagine que desea crear un sitio de rentas de películas en línea para competir con la tienda iTunes de Apple. Un sitio web regular permitirá a los usuarios descargar y luego ver los videos (después de que paguen, por supuesto). En la figura 7-50 se muestra la secuencia de pasos. Los describiremos para contrastarlos con el siguiente ejemplo.

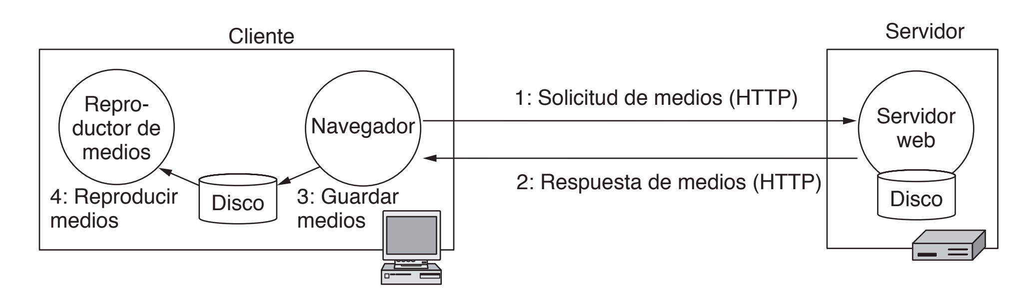


Figura 7-50. Reproducción de medios a través de la web mediante simples descargas.

El navegador entra en acción cuando el usuario hace clic en una película. En el paso 1, envía una solicitud HTTP por la película al servidor web al que la película está vinculada. En el paso 2, el servidor obtiene la película (que es sólo un archivo en MP4 o en algún otro formato) y la envía de vuelta al navegador. Mediante el uso del tipo MIME, por ejemplo, *video/mp4*, el navegador detecta cómo se supone que debe mostrar el archivo. En este caso es mediante un reproductor de medios que se muestra como una aplicación ayudante, aunque también podría ser un complemento. En el paso 3, el navegador guarda toda la película en un archivo de trabajo en el disco. Después inicia el reproductor de medios y le pasa el nombre del archivo de trabajo. Por último, en el paso 4 el reproductor de medios empieza a leer el archivo y reproduce la película.

En principio, esta metodología es totalmente correcta. Logrará reproducir la película. No hay problema de red en tiempo real con el que haya que lidiar, ya que la descarga es simplemente la descarga de un archivo. El único problema es que hay que transmitir todo el video a través de la red antes de que empiece la película. La mayoría de los clientes no desean esperar una hora a que se descargue su “video bajo demanda”. Este modelo puede ser problemático incluso para el audio. Imagine escuchar una muestra de una canción antes de comprar un álbum. Si la canción es de 4 MB, que es un tamaño común para una canción MP3, y la conectividad de banda ancha es de 1 Mbps, el usuario tendrá que esperar medio minuto en silencio hasta que empiece la muestra. Es poco probable que este modelo vaya a vender muchos álbumes.

Para resolver este problema sin modificar la forma en que funciona el navegador, los sitios pueden usar el diseño que se muestra en la figura 7-51. La página vinculada a la película no es el archivo de película verdadero, sino lo que se conoce como **metaarchivo**, un archivo muy corto que sólo da nombre a la película (y que posiblemente tiene otros descriptores clave). Un metaarchivo simple podría ser sólo una línea de texto ASCII y verse así:

```
rtsp://servidor-video-joe/película-0025.mp4
```

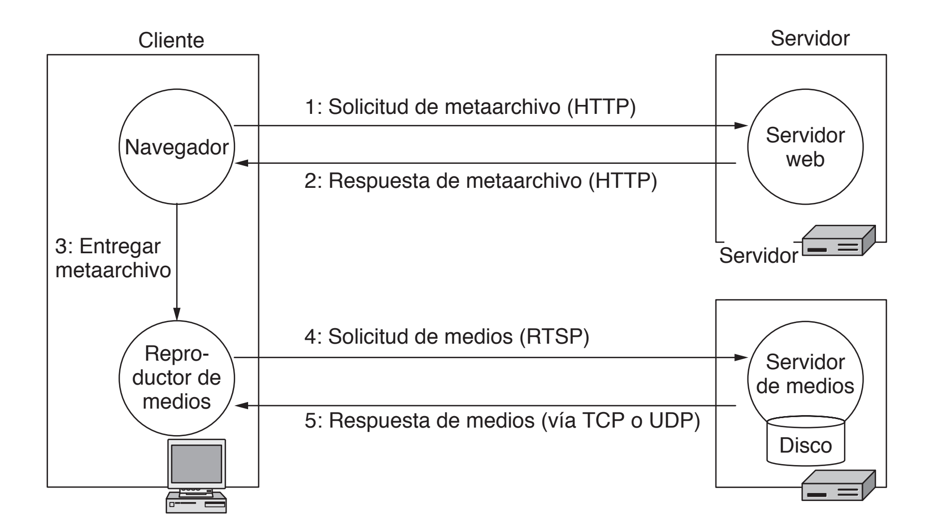


Figura 7-51. Transmisión de medios de flujo continuo mediante el uso de web y un servidor de medios.

El navegador obtiene la página de la manera usual (sólo que ahora es un archivo de una línea) en los pasos 1 y 2. Después inicia el reproductor de medios y le entrega el archivo de una línea en el paso 3, de la manera usual. El reproductor de medios lee el metaarchivo y ve el URL de donde puede obtener la película. Se contacta con el *servidor-video-joe* y le pide la película en el paso 4. A continuación la película se transmite por flujo continuo al reproductor de medios en el paso 5. La ventaja de este arreglo es que el reproductor de medios empieza rápidamente, sólo después de descargar un metaarchivo muy corto. Una vez que esto ocurre, el navegador queda fuera del ciclo. El medio se envía directamente al reproductor de medios, que puede empezar a mostrar la película antes de que se haya descargado por completo.

Mostramos dos servidores en la figura 7-51 debido a que, por lo general, el servidor que se menciona en el metaarchivo no es el mismo que el servidor web. De hecho, generalmente ni siquiera es un servidor HTTP, sino un servidor de medios especializado. En este ejemplo, el servidor de medios usa **RTSP** (**Protocolo de Flujo Continuo en Tiempo Real**, del inglés *Real Time Streaming Protocol*), como se indica mediante el nombre *rtsp* del esquema.

El reproductor de medios debe realizar cuatro tareas importantes:

1. Administrar la interfaz de usuario.
2. Manejar los errores de transmisión.
3. Descomprimir el contenido.
4. Eliminar la variación del retardo (*jitter*).

En la actualidad, la mayoría de los reproductores de medios tienen una interfaz de usuario ostentosa, que algunas veces simula una unidad estereofónica con botones, perillas, botones deslizables y pantallas visuales. A menudo hay paneles frontales intercambiables, conocidos como **skins**, que el usuario puede arrastrar hasta el reproductor. El reproductor de medios tiene que manejar todo esto e interactuar con el usuario.

Los otros trabajos son relacionados y dependen de los protocolos de red. Analizaremos cada uno de ellos en turno, empezando con el manejo de los errores de transmisión. El proceso de lidiar con los errores depende de si se utiliza un transporte basado en TCP para transportar los medios, como HTTP, o si se usa un transporte basado en UDP, tal como RTP. Ambos se utilizan en la práctica. Si se utiliza un transporte basado en TCP, entonces no hay errores que el reproductor de medios tenga que corregir, ya que TCP

provee confiabilidad de antemano mediante el uso de retransmisiones. Ésta es una manera sencilla de manejar los errores, al menos para el reproductor de medios, pero complica la eliminación de la variación del retardo (*jitter*) en un paso posterior.

Como alternativa se puede utilizar un transporte basado en UDP, como RTP, para desplazar los datos. Lo estudiamos en el capítulo 6. Con estos protocolos no hay retransmisiones. Así, la pérdida de paquetes debido a la congestión o los errores de transmisión significará que no llegarán algunos de los medios. Es responsabilidad del reproductor de medios lidiar con este problema.

Vamos a comprender la dificultad a la que nos enfrentamos. La pérdida es un problema, puesto que a los clientes no les gustan los grandes espacios vacíos en sus canciones o películas. Sin embargo, no es un problema tan grave como la pérdida en la transferencia de un archivo regular, ya que al perder una pequeña cantidad de medios no se degrada la presentación para el usuario. En el video es poco probable que el usuario se dé cuenta si hay 24 nuevas tramas en cierto segundo en vez de 25. En el audio, los espacios vacíos cortos en la reproducción se pueden enmascarar con sonidos cercanos en el tiempo. Es muy improbable que el usuario detecte esta sustitución, a menos que esté poniendo *mucha* atención.

Sin embargo, la clave para el razonamiento anterior es que los espacios vacíos sean muy cortos. Por lo general, la congestión en la red o un error de transmisión provocarán que se pierda todo un paquete completo, y los paquetes se pierden con frecuencia en pequeñas ráfagas. Se pueden usar dos estrategias para reducir el impacto de la pérdida de paquetes en los medios perdidos: FEC e intercalado. Enseguida describiremos cada una.

FEC (**Corrección de Errores hacia Adelante**, del inglés *Forward Error Correction*) es simplemente la codificación de corrección de errores que estudiamos en el capítulo 3, sólo que en este caso se emplea a nivel de aplicación. La paridad entre los paquetes provee un ejemplo (Shacham y McKenny, 1990). Por cada cuatro paquetes de datos que se envían, se puede construir y enviar un quinto **paquete de paridad**. Esto se muestra en la figura 7-52 con los paquetes *A*, *B*, *C* y *D*. El paquete de paridad *P* contiene bits redundantes que son la paridad o sumas con OR exclusivo de los bits en cada uno de los cuatro paquetes de datos. Con suerte, todos los paquetes llegarán para la mayoría de los grupos de cinco paquetes. Cuando esto ocurre, el paquete de paridad simplemente se descarta en el receptor. O, si sólo se pierde el paquete de paridad, no hubo ningún daño.

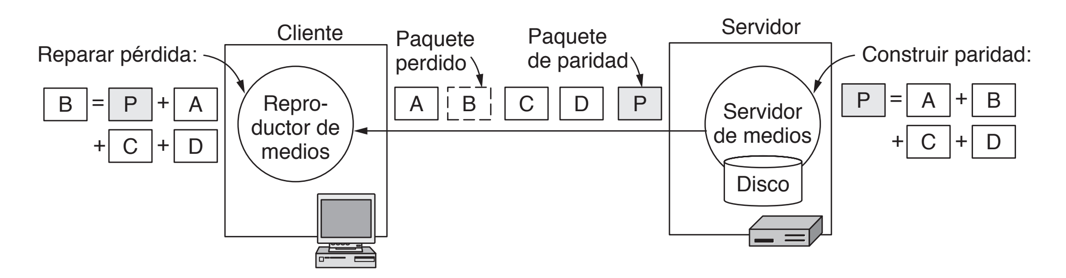


Figura 7-52. Uso de un paquete de paridad para reparar pérdidas.

Sin embargo, en ocasiones tal vez se pierda un paquete de datos durante la transmisión, como *B* en la figura 7-52. El reproductor de medios sólo recibe tres paquetes de datos: *A*, *C* y *D*, más el paquete de paridad *P*. Por diseño, los bits en el paquete de datos faltante se pueden reconstruir a partir de los bits de paridad. En específico, mediante el uso de "+" para representar la suma con OR exclusivo o módulo 2, podemos reconstruir *B* como $B = P + A + C + D$ mediante las propiedades del OR exclusivo (es decir, $X + Y + Y = X$).

La FEC puede reducir el nivel de pérdida visto por el reproductor de medios mediante la reparación de algunos de los paquetes perdidos, pero sólo es efectiva hasta cierto nivel. Si se pierden dos paquetes en un grupo de cinco, no hay nada que podamos hacer para recuperar los datos. La otra propiedad a tener en cuenta sobre la FEC es el costo que tenemos que pagar para obtener esta protección. Los grupos de cuatro paquetes se convierten en grupos de cinco por lo que los requerimientos de ancho de banda de los medios aumentan un 25%. La latencia de la decodificación también aumenta, ya que tal vez tengamos que esperar hasta que llegue el paquete de paridad para reconstruir un paquete de datos que haya llegado antes.

También hay un truco astuto en la técnica anterior. En el capítulo 3 describimos que la paridad provee detección de errores. Aquí proveemos corrección de errores. ¿Cómo puede hacer ambas cosas? La respuesta es que, en este caso, se sabe cuál paquete se perdió. Al conjunto de datos perdidos se le conoce como **borrado** (*erasure*). En el capítulo 3, cuando analizamos el caso en que una trama se recibe con ciertos bits erróneos, no sabíamos qué bits eran erróneos. Es más difícil lidiar con este caso que con el de los borrados. Así, cuando hay borrados la paridad puede proveer corrección de errores, y sin borrados la paridad sólo puede proveer detección de errores. Pronto veremos otro beneficio inesperado de la paridad, cuando analicemos los escenarios de multidifusión.

La segunda estrategia se conoce como **intercalado**. Esta metodología se basa en mezclar o intercalar el orden de los medios antes de la transmisión, y de separarlos al momento de recibirlos. De esa forma, si se pierde un paquete (o una ráfaga de paquetes), la pérdida se esparcirá a través del tiempo gracias a la separación. No se producirá un solo vacío extenso cuando se reproduzcan los medios. Por ejemplo, un paquete podría contener 220 muestras estereofónicas, cada una de las cuales puede contener un par de números de 16 bits, que por lo general está bien para 5 mseg de música. Si las muestras se enviaran en orden, un paquete perdido representaría un vacío de 5 mseg en la música. En cambio, las muestras se transmiten como se indica en la figura 7-53. Todas las muestras pares para un intervalo de 10 mseg se envían en un paquete, seguidas de todas las muestras impares en el siguiente. Ahora, la pérdida del paquete 3 no representa un vacío de 5 mseg en la música, sino la pérdida de cualquier otra muestra por 10 mseg. Esta pérdida se puede manejar con facilidad al hacer que el reproductor de medios realice una interpolación usando las muestras anterior y posterior. El resultado es una resolución temporal menor durante 10 mseg, y no un vacío de tiempo detectable en los medios.

Este esquema anterior de intercalado sólo funciona con el muestreo descomprimido. Sin embargo, el intercalado (durante periodos cortos, no con muestras individuales) también se puede aplicar después de la compresión, siempre y cuando haya una forma de encontrar límites de muestra en el flujo comprimido. El RFC 3119 proporciona un esquema que funciona con audio comprimido.

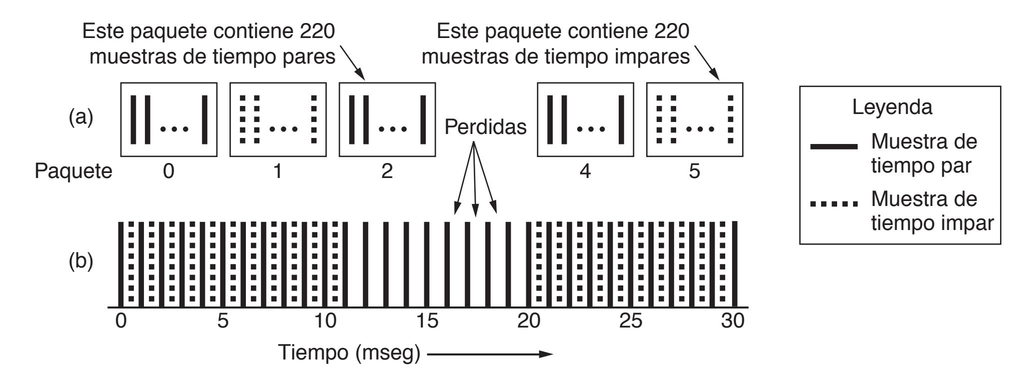


Figura 7-53. Cuando los paquetes transportan muestras alternas, la pérdida de un paquete reduce la resolución temporal en vez de crear un vacío en el tiempo.

El intercalado es una técnica atractiva cuando se puede utilizar, ya que no necesita de ancho de banda adicional, a diferencia de la FEC. Sin embargo, el intercalado aumenta la latencia, justo igual que la FEC, debido a la necesidad de esperar a que llegue un grupo de paquetes (para poderlos separar).

La tercera tarea del reproductor de medios es descomprimir el contenido. Aunque esta tarea requiere mucho poder de cómputo, es bastante simple. La cuestión complicada es cómo decodificar los medios si el protocolo de red no corrige los errores de transmisión. En muchos esquemas de compresión, los datos posteriores no pueden descomprimirse hasta que lo hayan hecho los datos anteriores, debido a que los datos posteriores se codifican de manera relativa a los datos anteriores. En un transporte basado en UDP, puede haber pérdida de paquetes. Por ende, hay que diseñar el proceso de codificación para permitir la decodificación a pesar de la pérdida de paquetes. Este requerimiento es la razón por la cual MPEG usa tramas I, P y B. Cada trama I se puede decodificar de manera independiente a las otras tramas, para recuperarse de la pérdida de cualquier trama anterior.

La cuarta tarea es eliminar la variación del retardo (*jitter*), la perdición de todos los sistemas de tiempo real. La solución general que describimos en la sección 6.4.3 es usar un búfer de reproducción. Todos los sistemas de flujo continuo empiezan por colocar en el búfer entre 5 y 10 segundos de medios antes de empezar a reproducirlos, como se muestra en la figura 7-54. La reproducción extrae medios del búfer de manera regular, de modo que el audio sea claro y el video sea uniforme. El retardo al inicio proporciona al búfer la oportunidad de llenar la **marca de nivel bajo**. La idea es que ahora los datos deben llegar con la frecuencia suficiente como para que el búfer nunca se vacíe por completo. Si ocurriera esto, el reproductor de medios quedaría paralizado. El valor del almacenamiento en búfer es que, si los datos algunas veces tardan en llegar debido a la congestión, los medios en el búfer permitirán que la reproducción continúe de la manera usual hasta que lleguen nuevos medios y se reabastezca el búfer.

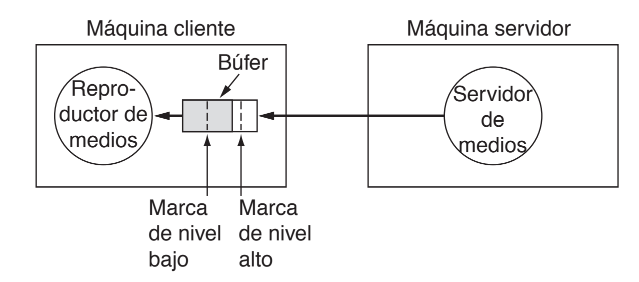


Figura 7-54. El reproductor de medios almacena en un búfer la entrada del servidor de medios y reproduce desde el búfer en vez de hacerlo directamente desde la red.

La cantidad necesaria de almacenamiento en búfer, además de la rapidez con que el servidor de medios envíe los recursos para llenar el búfer, dependen de los protocolos de red. Existen muchas posibilidades. El mayor factor en el diseño es si se utiliza un transporte basado en UDP o uno basado en TCP.

Suponga que se utiliza un transporte basado en UDP, como RTP. Suponga además que hay mucho ancho de banda para enviar paquetes desde el servidor de medios hasta el reproductor de medios con poca pérdida, y que hay poco tráfico adicional en la red. En este caso, los paquetes se pueden enviar a la misma tasa con la que se reproducen los medios. Cada paquete viajará por la red y, después de un retardo de propagación, llegará en el tiempo apropiado para que el reproductor de medios presente esos medios. Se requiere muy poco almacenamiento en búfer, ya que no hay variabilidad en el retardo. Si se utiliza intercalado o FEC, se requiere más almacenamiento en búfer para al menos el grupo de paquetes a través de los cuales se realice el intercalado o la FEC. Sin embargo, esto sólo agrega una pequeña cantidad de almacenamiento en búfer.

Por desgracia, este escenario no es realista en dos aspectos. En primer lugar, el ancho de banda varía a través de las rutas de red, por lo que en general no está claro para el servidor de medios si habrá o no suficiente ancho de banda antes de que intente transmitir los medios en flujo continuo. Una solución simple es codificar los medios a varias resoluciones y dejar que cada usuario seleccione una resolución soportada por su conexión a Internet. A menudo hay sólo dos niveles: alta calidad, con una codificación de 1.5 Mbps o mayor, y baja calidad, con una codificación de 512 kbps o menor.

En segundo lugar, habrá algo de variación en el retardo (*jitter*) o variación en cuanto al tiempo que tardan las muestras de los medios en cruzar la red. Esta variación en el retardo proviene de dos fuentes. Con frecuencia hay una cantidad considerable de tráfico competidor en la red (parte del cual puede provenir de los mismos usuarios multitarea que naveguen por la web mientras observan aparentemente una película de flujo continuo). Este tráfico producirá fluctuaciones cuando lleguen los medios. Además, nos importa la llegada de las tramas de video y las muestras de audio, no los paquetes. Con la compresión, las tramas de video en especial pueden ser más grandes o más pequeñas, dependiendo de su contenido. Una secuencia de acción por lo general requerirá más bits para codificarse que un paisaje plácido. Si el ancho de banda de la red es constante, la tasa de distribución de los medios contra el tiempo variará. Entre más variación haya en el retardo de estas fuentes, mayor tiene que ser la marca de nivel bajo del búfer para evitar insuficiencia de datos.

Ahora suponga que se utiliza un transporte basado en TCP, como HTTP, para enviar los medios. Al realizar retransmisiones y esperar a entregar los paquetes hasta que estén en orden, TCP incrementará la variación en el retardo que observa el reproductor de medios, tal vez de manera considerable. El resultado es que se necesita un búfer más grande y una marca de nivel bajo más alta. Sin embargo, hay una ventaja. TCP enviará datos tan rápido como la red pueda transportarlos. Algunas veces los medios se pueden retardar si hay que reparar la pérdida. Pero gran parte del tiempo, la red podrá entregar los medios con más rapidez de la que el reproductor los consuma. En estos periodos, el búfer se llenará y evitará insuficiencias de datos posteriores. Si la red es mucho más rápida que la tasa promedio de medios, como es comúnmente el caso, el búfer se llenará con rapidez después del inicio de la transmisión, de tal forma que pronto deje de ser una preocupación el hecho de que se llegue a vaciar.

Con TCP, o con UDP y una tasa de transmisión que sea mayor a la tasa de reproducción, una duda es qué tan lejos del punto de reproducción están dispuestos a proceder el reproductor y el servidor de medios. Lo común es que estén dispuestos a descargar todo el archivo.

Sin embargo, el hecho de proceder mucho más allá del punto de reproducción implica realizar trabajo que no se necesita aún, tal vez se requiera un espacio de almacenamiento considerable y no sea necesario para evitar insuficiencias de datos del búfer. Cuando no se desea, la solución es que el reproductor de medios defina una **marca de nivel alto** en el búfer. En esencia, el servidor simplemente inyecta datos hasta que el búfer se llene a la marca del nivel alto. Después, el reproductor de medios le indica que se detenga. Como los datos seguirán entrando hasta que el servidor haya obtenido la solicitud de pausa, la distancia entre la marca de nivel alto y el final del búfer tiene que ser mayor que el producto ancho de banda-retardo de la red. Una vez que el servidor se detenga, el búfer empezará a vaciarse. Cuando llegue a la marca de nivel bajo, el reproductor de medios indicará al servidor de medios que empiece a enviar de nuevo. Para evitar la insuficiencia de datos, la marca de nivel bajo también debe tener en cuenta el producto ancho de banda-retardo de la red a la hora de pedir al servidor de medios que reanude el envío de los medios.

Para iniciar y detener el flujo de los medios, el reproductor de medios necesita un control remoto para ello. Esto es lo que RTSP provee. Se define en el RFC 2326 y provee el mecanismo para que el reproductor controle al servidor. Además de iniciar y detener el flujo de medios, puede buscar hacia atrás o hacia adelante cierta posición, reproducir intervalos específicos y a velocidades rápidas o lentas. Pero no se encarga del flujo de datos, que por lo general es RTP sobre UDP, o RTP sobre HTTP sobre TCP.

En la figura 7-55 se listan los comandos principales que proporciona RTSP. Tienen un formato de texto simple, como los mensajes HTTP, y por lo general se transportan a través de TCP. RTSP puede operar a través de UDP también, ya que se confirma la recepción de cada comando (y, por lo tanto, se puede reenviar si no se confirma).

Comando	Acción del servidor
DESCRIBE	Listar los parámetros de medios.
SETUP	Establecer un canal lógico entre el reproductor y el servidor.
PLAY	Empezar a enviar datos al cliente.
RECORD	Empezar a aceptar datos del cliente.
PAUSE	Dejar temporalmente de enviar datos.
TEARDOWN	Liberar el canal lógico.

Figura 7-55. Comandos RTSP del reproductor para el servidor.

Aun cuando TCP parecería no adaptarse bien al tráfico en tiempo real, a menudo se utiliza en la práctica. La principal razón es que puede pasar a través de los *firewalls* con más facilidad que UDP, en especial cuando se ejecuta a través del puerto HTTP. La mayoría de los administradores configuran los *firewalls* para proteger sus redes de los visitantes no deseados. Casi siempre permiten que las conexiones TCP del puerto remoto 80 pasen para el tráfico HTTP y web. Si se bloquea ese puerto, pronto habrá muchos usuarios inconformes. Sin embargo, la mayoría de los otros puertos sí se bloquean, incluyendo los de RTSP y RTP, que utilizan los puertos 554 y 5 004, entre otros. Así, la forma más fácil de obtener medios de flujo continuo a través del *firewall* es que el sitio web pretenda que es un servidor HTTP que envía una respuesta HTTP regular, por lo menos para el *firewall*.

Hay algunas otras ventajas de usar TCP. Como provee confiabilidad, TCP otorga al cliente una copia completa de los medios. Esto hace que sea fácil para un usuario rebobinar hasta un punto de reproducción visto con anterioridad, sin preocuparse por la pérdida de datos. Finalmente, TCP almacenará en el búfer tantos medios como pueda y con la mayor rapidez posible. Cuando el espacio en el búfer es económico (que es cuando se usa el disco para el almacenamiento), el reproductor de medios puede descargarlos mientras el usuario observa. Una vez completa la descarga, el usuario puede observar sin interrupciones, incluso aunque pierda la conectividad. Esta propiedad es útil para los móviles, ya que la conectividad puede cambiar rápidamente con el movimiento.

La desventaja de TCP es la latencia adicional al inicio (debido al inicio de TCP), además de una marca de nivel bajo más alta. Sin embargo, raras veces es esto un castigo, siempre y cuando el ancho de banda de la red sea mucho mayor que la tasa de los medios.

7.4.4 Transmisión en flujo continuo de medios en vivo

No son sólo los videos grabados los que tienen una enorme popularidad en la web. La transmisión en flujo continuo de medios en vivo también es muy popular. Una vez que fue posible transmitir en flujo continuo audio y video a través de Internet, las estaciones comerciales de radio y TV tuvieron la idea de difundir su contenido a través de Internet, además de hacerlo por el aire. No pasó mucho tiempo después de ello cuando las estaciones universitarias empezaron a transmitir sus señales por Internet. Después los *estudiantes* universitarios empezaron sus propias difusiones por Internet.

En la actualidad, personas y empresas de todos tamaños transmiten en flujo continuo audio y video en vivo. El área es un foco de innovación a medida que evolucionan las tecnologías y los estándares. Las principales estaciones de televisión utilizan la transmisión de flujo continuo en vivo para una presencia en línea. A esto se le conoce como **IPTV** (**Televisión IP**, del inglés *IP TeleVision*). También se utiliza para difundir estaciones de radio como BBC. A esto se le conoce como **radio por Internet**. Tanto la IPTV como la radio por Internet llegan a audiencias mundiales a través de eventos que varían, desde exposiciones de moda hasta la Copa Mundial de futbol y partidos de prueba transmitidos en vivo desde el Melbourne Cricket Ground. Los proveedores de cable utilizan la transmisión de flujo continuo en vivo sobre IP como una tecnología en la que basan sus propios sistemas de difusión. Y las operaciones de bajo presupuesto también la utilizan, desde los sitios para adultos hasta los zoológicos. Con la tecnología actual, prácticamente cualquiera puede empezar a transmitir flujos continuos en vivo sin muchas trabas y con pocos gastos.

Una metodología para transmitir flujo continuo en vivo es guardar los programas en disco. Los espectadores se pueden conectar a los archivos del servidor, localizar un programa y descargarlo para escucharlo. Un **podcast** es un episodio que se recupera de esta manera. En cuanto a los eventos programados, también es posible almacenar contenido justo después de transmitirlo en vivo, de modo que el archivo sólo esté operando, por decir, a media hora o menos de retraso de la transmisión en vivo.

De hecho, esta metodología es exactamente igual a la que se utiliza para los medios de flujo continuo que acabamos de ver. Es fácil de hacer, todas las técnicas que hemos analizado funcionan para esta metodología y los espectadores pueden elegir entre todos los programas del archivo.

Una metodología distinta es difundir en vivo a través de Internet. Los espectadores sintonizan un flujo continuo de medios que se está transmitiendo en ese momento, justo igual como encender la televisión. Sin embargo, los reproductores de medios proveen las características adicionales que permiten al usuario detener o rebobinar la reproducción. Los medios en vivo continuarán transmitiéndose en flujo continuo y se almacenarán en el búfer del reproductor hasta que el usuario esté listo para verlos. Desde el punto de vista del navegador, es exactamente igual al caso en el que se transmiten por flujo continuo los medios almacenados. Al reproductor no le importa si el contenido proviene de un archivo o se está transmitiendo en vivo, y por lo general no lo podrá detectar (excepto que no es posible saltar hacia delante en un flujo continuo en vivo). Dada la similitud del mecanismo, se aplica gran parte de nuestro análisis anterior, pero también hay algunas diferencias clave.

Un punto importante es que todavía existe la necesidad de almacenar en un búfer del lado cliente para suavizar la variación en el retardo. De hecho, a menudo se requiere una mayor cantidad de almacenamiento en el búfer para los flujos continuos en vivo (sin importar el hecho de que el usuario puede detener la reproducción). Cuando se transmite un flujo continuo desde un archivo, los medios se pueden enviar a una tasa mayor que la tasa de reproducción. Esto hará que se acumule rápidamente un búfer para compensar la variación del retardo en la red (y el reproductor detendrá el flujo si no quiere almacenar más datos en el búfer). En contraste, los flujos continuos de medios en vivo siempre se transmiten a la tasa que se generan, que es igual a la tasa a la que se reproducen. No se pueden enviar más rápido. Como consecuencia, el búfer debe ser lo bastante grande como para manejar el rango completo de variación del retardo. En la práctica, un retardo inicial de 10 a 15 segundos es generalmente adecuado, por lo que esto no representa un grave problema.

La otra diferencia importante es que, por lo general, los eventos de flujos continuos en vivo tienen cientos o miles de espectadores simultáneos para el mismo contenido. Bajo estas circunstancias, la solución natural para los flujos continuos en vivo es usar la multidifusión. Éste no es el caso para transmitir flujo continuo de medios almacenados, ya que por lo común los usuarios transmiten por flujo continuo distinto contenido en cualquier momento dado. Así, la transmisión en flujo continuo para muchos usuarios consiste en muchas sesiones de flujo continuo individuales, que ocurren al mismo tiempo.

Un esquema de transmisión en flujo continuo por multidifusión funciona de la siguiente manera. El servidor envía cada paquete de medios una vez usando multidifusión IP a una dirección de grupo. La red entrega una copia del paquete a cada miembro del grupo. Todos los clientes que desean recibir el flujo se han unido al grupo. Para ello utilizan IGMP en vez de enviar un mensaje RTSP al servidor de medios. Esto se debe a que el servidor de medios ya está enviando el flujo continuo en vivo (excepto antes de que se una el primer usuario). Lo que se necesita es hacer los arreglos necesarios para que el flujo continuo se reciba en forma local.

Como la multidifusión es un servicio de entrega de uno a muchos, los medios se transportan en paquetes RTP a través de un transporte UDP. TCP sólo opera entre un solo emisor y un solo receptor. Como UDP no provee confiabilidad, tal vez se pierdan algunos paquetes. Para reducir el nivel de pérdida de medios a un nivel aceptable podemos usar FEC e intercalado, como antes.

En el caso de la FEC, hay una interacción benéfica con la multidifusión que se muestra en el ejemplo de paridad de la figura 7-56. Cuando los paquetes se transmiten por multidifusión, los distintos clientes pueden perder distintos paquetes. Por ejemplo, el cliente 1 perdió el paquete *B*, el cliente 2 perdió el paquete de paridad *P*, el 3 perdió *D* y el 4 no perdió ningún paquete. Sin embargo, aun cuando se perdieron tres paquetes distintos entre los clientes, cada uno puede recuperar todos los paquetes de datos en este ejemplo. Todo lo que se requiere es que cada cliente no pierda más de un paquete, sin importar cuál sea, de modo que el paquete faltante se pueda recuperar mediante un cálculo de paridad. Nonnenmacher y sus colaboradores (1997) describen cómo se puede usar esta idea para aumentar la confiabilidad.

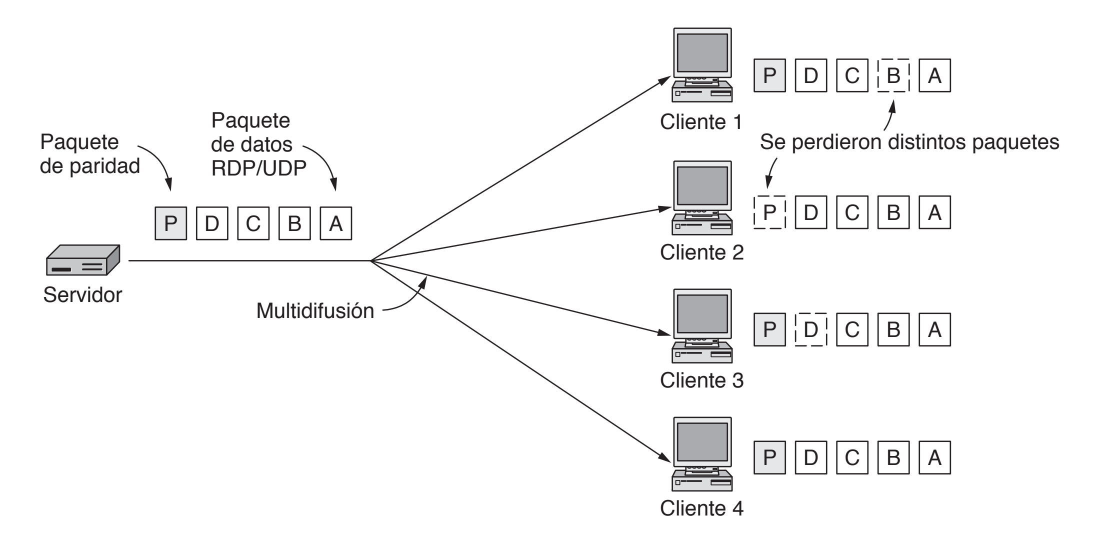


Figura 7-56. Multidifusión de medios de flujo continuo con un paquete de paridad.

En un servidor con un gran número de clientes, la multidifusión de medios en paquetes RTP y UDP es sin duda la manera más eficiente de operar. De cualquier modo, el servidor debe transmitir *N* flujos cuando tiene *N* clientes, para lo cual se requerirá una cantidad muy grande de ancho de banda de red en el servidor para los grandes eventos de flujo continuo.

Tal vez le sorprenda saber que Internet no funciona así en la práctica. Lo que ocurre por lo general es que cada usuario establece una conexión TCP separada con el servidor, y los medios se transmiten en flujo continuo a través de esa conexión. Para el cliente, esto es lo mismo que transmitir medios almacenados en flujo continuo. Y al igual que la transmisión en flujo continuo de medios almacenados, hay varias razones por las que ésta es aparentemente una mala elección.

La primera razón es que la multidifusión IP no está ampliamente disponible en Internet. Algunos ISP y redes la soportan de manera interna, pero por lo general no está disponible entre los límites de las redes, como se requiere para los flujos continuos de área amplia. Las otras razones son las mismas ventajas de TCP sobre UDP, como vimos antes. La transmisión de flujo continuo mediante TCP llegará casi a todos los clientes en Internet, en especial cuando se disfrace como HTTP para pasar a través de los *firewalls*, y la entrega de medios confiable permite a los usuarios rebobinar con facilidad.

No obstante, hay un caso importante en el que se pueden usar UDP y la multidifusión para transmitir flujos continuos: dentro de una red de proveedor. Por ejemplo, tal vez una empresa de cable decida difundir canales de TV a los receptores digitales (*set-top box*) de los clientes mediante el uso de tecnología IP en vez de las difusiones de video tradicionales. El uso de IP para distribuir video de difusión se conoce generalmente como IPTV, como vimos antes. Puesto que la compañía de cable tiene todo el control sobre su propia red, puede manipularla para soportar multidifusión IP y tener suficiente ancho de banda para la distribución basada en UDP. Todo esto es invisible para el cliente, ya que la tecnología IP existe dentro del **jardín cercado** del proveedor. Parece similar a la TV por cable en términos de servicio, pero en el fondo es IP, en donde el receptor digital es una computadora que ejecuta UDP y el televisor es simplemente un monitor conectado a la computadora.

Ahora regresemos al caso de Internet. La desventaja de transmitir flujo continuo en vivo sobre TCP es que el servidor debe enviar una copia separada de los medios para cada cliente. Esto es factible para un número moderado de clientes, en especial para el audio. El truco es colocar el servidor en una ubicación con buena conectividad a Internet, de modo que haya suficiente ancho de banda. Por lo general esto significa rentar un servidor en un centro de datos de un proveedor de hospedaje, no usar un servidor en el hogar sólo con conectividad a Internet de banda ancha. Hay un mercado de hospedaje web competitivo, por lo que no necesita ser algo costoso.

De hecho, es fácil para cualquiera (incluso un estudiante) establecer y operar un servidor de medios de flujo continuo, como una estación de radio por Internet. En la figura 7-57 se muestran los principales componentes de esta estación. La base de la estación es una PC ordinaria con una tarjeta de sonido y un micrófono decentes. Se utiliza software popular para capturar audio y codificarlo en varios formatos (por ejemplo, MP4) y se utilizan reproductores de medios para escuchar el audio de la manera usual.

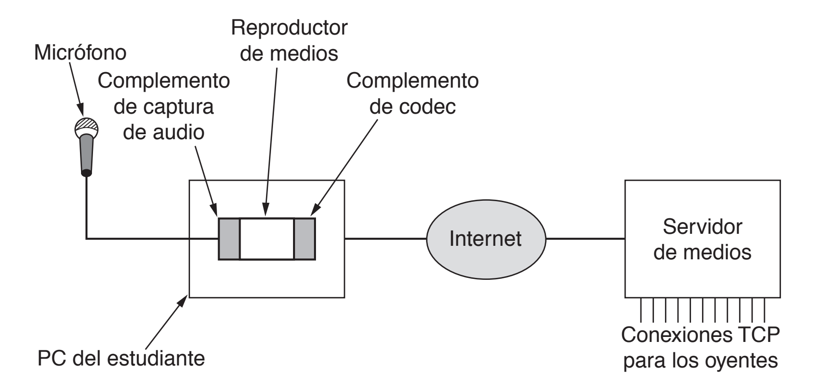


Figura 7-57. Estación de radio de un estudiante.

El flujo de audio capturado en la PC se alimenta después a través de Internet hasta un servidor de medios con buena conectividad de red, ya sea en forma de podcasts para transmitir por flujo continuo el archivo almacenado, o para la transmisión del flujo continuo en vivo. El servidor se encarga de la tarea de distribuir los medios a través de grandes cantidades de conexiones TCP. También presenta un sitio web de *front-end* con páginas sobre la estación y vínculos al contenido disponible para la transmisión de

flujo continuo. Hay paquetes de software comerciales para administrar todas las piezas, así como paquetes de código fuente abierto, como *icecast*.

Sin embargo, para una cantidad muy grande de clientes, se vuelve imposible usar TCP para enviar medios a cada cliente desde un solo servidor. Simplemente no hay suficiente ancho de banda para el único servidor. En sitios grandes, la transmisión de flujo continuo se realiza mediante un conjunto de servidores distribuidos en forma geográfica, de modo que un cliente se pueda conectar al servidor más cercano. Ésta es una red de distribución de contenido que estudiaremos al final del capítulo.

7.4.5 Conferencia en tiempo real

Al principio las llamadas de voz se transmitían a través de la red telefónica pública conmutada, y el tráfico de red era principalmente tráfico de voz, con un poco de tráfico de datos de vez en cuando. Después llegaron Internet y la web. El tráfico de datos creció y creció, hasta que en 1999 había tanto tráfico de datos como de voz (como ahora la voz es digitalizada, ambos tipos de tráfico se pueden medir en bits). Para el 2002, el volumen del tráfico de datos era 10 veces más que el volumen del tráfico de voz y seguía creciendo en forma exponencial, mientras que el tráfico de voz permanecía casi constante.

La consecuencia de este crecimiento ha sido que la red telefónica dio un giro de 180 grados. Ahora el tráfico de voz se transmite mediante el uso de tecnologías de Internet y representa sólo una pequeña fracción del ancho de banda de la red. Esta tecnología perjudicial se conoce como **voz sobre IP** y también **telefonía de Internet**.

La voz sobre IP se usa en varias formas que son controladas por fuertes factores económicos (traducción al inglés: como ahorra dinero, la gente la utiliza). Una forma es tener lo que parecen teléfonos comunes (¿anticuados?) que se conectan a la Ethernet y envían llamadas a través de la red. Pehr Anderson era un estudiante universitario en el MIT cuando él y sus amigos construyeron el prototipo de este diseño para el proyecto de una clase. Recibieron una calificación de "B". Como no quedó contento, inició una empresa llamada NBX en 1996, fue el primero en trabajar con este tipo de voz sobre IP y lo vendió a 3Com por $90 millones tres años después. A las empresas les encanta este método, ya que les permite deshacerse de las líneas telefónicas separadas y aprovechan las redes que ya tienen.

Otra metodología es usar la tecnología IP para construir una red telefónica de larga distancia. En países como Estados Unidos, se puede acceder a esta red para obtener un servicio competitivo de larga distancia al marcar un prefijo especial. Las muestras de voz se colocan en paquetes que se inyectan a la red y se extraen de los paquetes cuando dejan la red. Como el equipo IP es mucho más económico que el equipo de telecomunicaciones, el resultado se traduce en servicios más económicos.

Como observación adicional, la diferencia en el precio no es del todo técnica. Por muchas décadas, el servicio telefónico fue un monopolio regulado que garantizaba a las compañías telefónicas una ganancia de un porcentaje fijo sobre sus costos. No es sorprendente que esto les llevara a incrementar los costos, por ejemplo, al tener mucho hardware redundante, lo cual se justificaba en nombre de una mejor confiabilidad (el sistema telefónico sólo podía estar caído durante un total de 2 horas cada 40 años, o 3 minutos cada año en promedio). Con frecuencia, a este efecto se le conocía como el "síndrome del poste telefónico chapado en oro". Desde la liberalización, sin duda el efecto ha disminuido, pero el equipo heredado todavía existe. La industria de la TI nunca tuvo un historial de operación como éste, puesto que siempre ha sido una industria optimizada y eficiente.

Sin embargo, nos concentraremos en la forma de voz sobre IP que tal vez sea la más visible para los usuarios: usar una computadora para llamar a otra computadora. Esta forma se volvió común a medida que las PC empezaron a venderse con micrófonos, bocinas, cámaras y CPUs con la suficiente rapidez como para procesar los medios, y las personas empezaron a conectarse a Internet desde su hogar con tasas de banda ancha. Un ejemplo muy conocido es el software Skype que se liberó a principios de 2003.

Skype y otras empresas también proveen puertas de enlace para facilitar el proceso de llamar a números telefónicos regulares, así como a computadoras con direcciones IP.

A medida que aumentaba el ancho de banda de red, las llamadas de video se unieron a las llamadas de voz. Al principio las llamadas de video se encontraban en el dominio de las empresas. Los sistemas de videoconferencias estaban diseñados para intercambiar video entre dos o más ubicaciones, lo cual permitiría a los ejecutivos en distintas ubicaciones verse mientras sostenían sus reuniones. Sin embargo, con una buena conectividad a Internet de banda ancha y un buen software de compresión de video, ahora los usuarios domésticos también pueden realizar videoconferencias. Las herramientas como Skype, que empezaron sólo con audio, ahora incluyen video con las llamadas, de modo que los amigos y familiares en todo el mundo se pueden ver y escuchar unos a otros.

Desde nuestro punto de vista, las llamadas de voz o video por Internet son también un problema de flujo continuo de medios, sólo que está mucho más restringido que el caso en que se transmite por flujo continuo un archivo almacenado o un evento en vivo. La restricción adicional es la baja latencia requerida para una conversación de dos vías. La red telefónica permite una latencia de una vía de hasta 150 mseg para un uso aceptable, después de lo cual el retardo empieza a percibirse como algo molesto para los participantes (las llamadas internacionales pueden tener una latencia de hasta 400 mseg, punto en el cual están lejos de ser una experiencia agradable para el usuario).

Esta baja latencia es difícil de lograr. Sin duda, almacenar en el búfer de cinco a 10 segundos de medios no va a funcionar (como funcionaría en la difusión de un evento deportivo en vivo). En cambio, los sistemas de video y voz sobre IP se deben rediseñar con una variedad de técnicas para minimizar la latencia. Este objetivo significa empezar con UDP como la opción clara en vez de TCP, ya que las retransmisiones de TCP introducen por lo menos un retardo con un valor de un viaje de ida y vuelta. Sin embargo, algunas formas de latencia no se pueden reducir, ni siquiera con UDP. Por ejemplo, la distancia entre Seattle y Amsterdam es de aproximadamente 8 000 km. El retardo de propagación de la velocidad de la luz para esta distancia en fibra óptica es de 40 mseg. Le deseamos buena suerte al que trate de superar eso. En la práctica, el retardo de propagación a través de la red será más largo, ya que cubrirá una distancia mayor (los bits no siguen una ruta de un gran círculo) y habrá retardos de transmisión a medida que cada enrutador IP almacene y reenvíe un paquete. Este retardo fijo consume parte del presupuesto para un retardo aceptable.

Otra fuente de latencia se relaciona con el tamaño de los paquetes. Por lo general, los paquetes grandes son la mejor forma de usar el ancho de banda de la red, ya que son más eficientes. Sin embargo, con una tasa de muestreo de audio de 64 kbps, un paquete de 1 KB tardaría 125 mseg en llenarse (e incluso más, si las muestras están comprimidas). Este retardo consumiría la mayor parte del presupuesto para el retardo en general. Además, si el paquete de 1 KB se envía a través de un enlace de acceso de banda ancha que opere a sólo 1 Mbps, tardará 8 mseg en transmitirse. Después hay que agregar otros 8 mseg para que el paquete pase a través del enlace de banda ancha en el otro extremo. Sin duda, los paquetes grandes no funcionarán.

En cambio, los sistemas de voz sobre IP usan paquetes cortos para reducir la latencia al costo de la eficiencia del ancho de banda. Agrupan las muestras de audio en unidades más pequeñas, comúnmente de 20 mseg. A 64 kbps, esto representa 160 bytes de datos, y menos con compresión. Sin embargo, por definición, el retardo de esta paquetización será de 20 mseg. El retardo de transmisión será menor también, ya que el paquete es más corto. En nuestro ejemplo, se reduciría a aproximadamente 1 mseg. Al usar paquetes cortos, se ha reducido el retardo mínimo de una vía para un paquete de Seattle a Amsterdam, de un valor inaceptable de 181 mseg (40 + 125 + 16) a un valor aceptable de 62 mseg (40 + 20 + 2).

Aún no hemos hablado de la sobrecarga de software, pero también consumirá una parte del presupuesto de retardo. Esto es en especial cierto para el video, ya que por lo general se necesita la compresión para ajustar el video al ancho de banda disponible. A diferencia de la transmisión por flujo continuo de un

archivo almacenado, no hay tiempo de usar un codificador con mucho poder de cómputo para obtener altos niveles de compresión. El codificador y el decodificador deben operar rápidamente.

De todas formas se requiere el almacenamiento en búfer para reproducir las muestras de medios a tiempo (para evitar el audio incomprensible o el video entrecortado), pero la cantidad de almacenamiento en el búfer se debe mantener muy pequeña, dado que el tiempo restante en nuestro presupuesto de retardo se mide en milisegundos. Cuando un paquete tarda demasiado en llegar, el reproductor omite las muestras faltantes y tal vez reproduzca sonido ambiental, o repita una trama para enmascarar la pérdida al usuario. Hay que ceder entre el tamaño del búfer que se utiliza para manejar la variación del retardo y la cantidad de medios que se pierde. Un búfer más pequeño reduce la latencia pero produce una mayor pérdida, debido a la variación del retardo. En un momento dado, a medida que se reduzca el tamaño del búfer, el usuario empezará a notar la pérdida.

Los lectores observadores podrán haber notado que no hemos dicho nada sobre los protocolos de capa de red hasta ahora en esta sección. La red puede reducir la latencia, o por lo menos la variación del retardo, mediante el uso de mecanismos de calidad del servicio. La razón de que esta cuestión no haya surgido antes es que la transmisión del flujo continuo puede operar con una latencia considerable, incluso en el caso del flujo continuo en vivo. Si la latencia no es una cuestión importante, basta con tener un búfer en el host final para lidiar con el problema de la variación del retardo. Sin embargo, en las conferencias en tiempo real, por lo general es importante que la red reduzca el retardo y la variación del mismo para cumplir con el presupuesto del retardo. La única situación en que no es importante es cuando hay tanto ancho de banda que todos reciben un buen servicio.

En el capítulo 5 describimos dos mecanismos de calidad del servicio que ayudan con este objetivo. Un mecanismo es **DS** (**Servicios Diferenciados**, del inglés *Differentiated Services*), en el cual los paquetes se marcan como pertenecientes a distintas clases que se manejan de manera distinta dentro de la red. La marca apropiada para los paquetes de voz sobre IP es retardo bajo. En la práctica, los sistemas establecen el punto de código DS con el valor conocido para la clase de *Reenvío expedito* con el tipo de servicio *Retardo bajo*. Esto es muy útil en los enlaces de acceso de banda ancha, ya que estos enlaces tienden a congestionarse cuando el tráfico web u otro tipo de tráfico compiten por el uso del enlace. Dada una ruta de red estable, el retardo y la variación del mismo aumentan por la congestión. Cada paquete de 1 KB tarda 8 mseg en enviarse a través de un enlace de 1 Mbps, y un paquete de voz sobre IP incurrirá en estos retardos si espera en una cola detrás del tráfico web. Sin embargo, con una marca de retardo bajo, los paquetes de voz sobre IP saltarán a la cabeza de la cola, pasando por alto a los paquetes web y reduciendo su retardo.

El segundo mecanismo que puede reducir el retardo es asegurar que haya suficiente ancho de banda. Si varía el ancho de banda disponible, o si hay una fluctuación en la tasa de transmisión (como con el video comprimido) y algunas veces no hay suficiente ancho de banda, las colas se llenarán y aumentará el retardo. Esto ocurrirá incluso con DS. Para asegurar un ancho de banda suficiente, se puede hacer una reservación con la red. Esta herramienta la proporcionan los servicios integrados. Por desgracia, no se implementa mucho. En cambio, las redes están diseñadas para un nivel de tráfico esperado, o a los clientes de red se les proporcionan acuerdos a nivel de servicio para un nivel de tráfico dado. Las aplicaciones deben operar debajo de este nivel para evitar producir congestión y evitar introducir retardos innecesarios. Para las videoconferencias ocasionales en el hogar, el usuario puede elegir una calidad de video como un proxy para las necesidades de ancho de banda, o el software puede probar la trayectoria de red y seleccionar una calidad apropiada de manera automática.

Cualquiera de los factores anteriores puede ocasionar que la latencia se vuelva inaceptable, por lo que en las conferencias en tiempo real es imprescindible poner atención a todos ellos. Para ver las generalidades de la tecnología de voz sobre IP y un análisis de estos factores, consulte a Goode (2002).

Ahora que hemos analizado el problema de la latencia en la transmisión de medios de flujo continuo, pasaremos al otro problema principal con el que deben lidiar los sistemas de conferencias. Este problema

es cómo establecer y terminar las llamadas. Examinaremos dos protocolos que se utilizan mucho para este fin: H.323 y SIP. Skype es otro sistema importante, pero los detalles sobre su funcionamiento interno son propietarios.

H.323

Algo que todos tenían claro antes de que se hicieran llamadas de voz y video a través de Internet era que, si cada distribuidor diseñaba su propia pila de protocolos, el sistema nunca funcionaría. Para evitar este problema, varias partes interesadas se reunieron bajo los auspicios de la ITU para desarrollar estándares. En 1996, la ITU emitió la recomendación **H.323** titulada como "Sistemas y equipo de telefonía visual para redes de área local que proveen una calidad de servicio no garantizada". Sólo la industria telefónica sería capaz de idear un nombre así. Éste se cambió rápidamente a "Sistemas de comunicaciones multimedia basados en paquetes" en la revisión de 1998. La recomendación H.323 fue la base para los primeros sistemas de conferencias por Internet que se hicieron muy populares. Sigue siendo la solución más implementada en la actualidad, en su séptima versión a partir de 2009.

H.323 es más una visión general arquitectónica de la telefonía de Internet que un protocolo específico. Hace referencia a una gran cantidad de protocolos específicos para la codificación de voz, el establecimiento de llamadas, la señalización, el transporte de datos y otras áreas, en vez de especificar estas cosas por su cuenta. En la figura 7-58 se ilustra el modelo general. En el centro hay una **puerta de enlace** que conecta a Internet con la red telefónica. Utiliza los protocolos H.323 del lado de Internet y los protocolos PSTN del lado telefónico. Los dispositivos de comunicación se llaman **terminales**. Una LAN puede tener un **guardián** (*gatekeeper*), el cual controla los puntos terminales bajo su jurisdicción, a lo cual se le denomina **zona**.

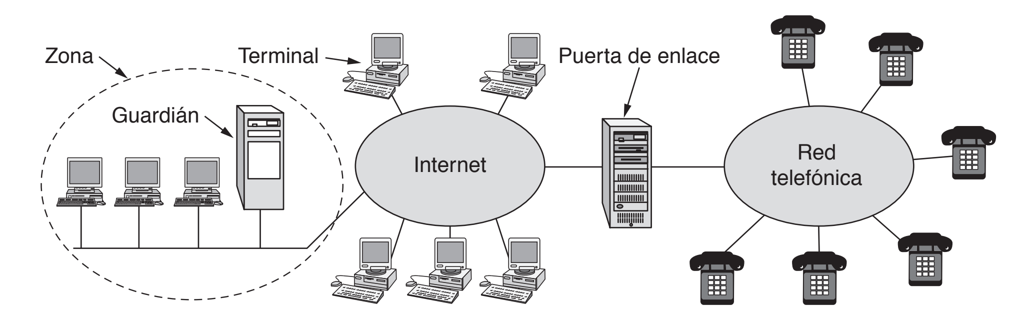


Figura 7-58. El modelo arquitectónico H.323 para la telefonía de Internet.

Una red telefónica necesita varios protocolos. Para empezar, hay un protocolo para codificar y decodificar audio y video. Las representaciones de la telefonía estándar de un solo canal de voz como 64 kbps de audio digital (8 000 muestras de 8 bits por segundo) se definen en la recomendación **G.711** de la ITU. Todos los sistemas H.323 deben soportar G.711. Se permiten otras codificaciones que comprimen la voz, pero no son requeridas. Usan distintos algoritmos de compresión y ofrecen distintas compensaciones entre calidad y ancho de banda. Para el video, se soportan las formas MPEG de compresión que describimos antes, incluyendo H.264.

Como se permiten varios algoritmos de compresión, se requiere un protocolo para que las terminales puedan negociar cuál van a usar. Este protocolo se conoce como **H.245**. También negocia otros aspectos de la conexión, como la tasa de bits. Se requiere RTCP para el control de los canales RTP. También se requiere un protocolo para establecer y liberar las conexiones, proveer tonos de marcado, hacer sonidos

de timbres y el resto de la telefonía estándar. Aquí se utiliza la recomendación **Q.931** de la ITU. Las terminales necesitan también un protocolo para hablar con el guardián (si está presente). Para este fin se utiliza **H.225**. El canal de PC a guardián que administra se conoce como canal **RAS** (**Registro**/**Admisión**/**Situación**, del inglés *Registration/Admisión/Status*). Este canal permite a las terminales unirse a la zona y salir de ella, solicitar y devolver ancho de banda, y proveer actualizaciones de estado, entre otras cosas. Finalmente, se necesita un protocolo para la transmisión real de los datos. Para este propósito se utiliza RTP sobre UDP. Se administra mediante RTCP, como siempre. En la figura 7-59 se muestra el posicionamiento de todos estos protocolos.

<table>
<tr><td>Audio</td><td>Video</td><td colspan="4">Control</td></tr>
<tr><td>G.7xx</td><td>H.26x</td><td rowspan="2">RTCP</td><td rowspan="2">H.225
(RAS)</td><td rowspan="2">Q.931
(Señalización)</td><td rowspan="2">H.245
(Control
de
llamadas)</td></tr>
<tr><td colspan="2">RTP</td></tr>
<tr><td colspan="4">UDP</td><td colspan="2">TCP</td></tr>
<tr><td colspan="6">IP</td></tr>
<tr><td colspan="6">Protocolo de capa de enlace</td></tr>
<tr><td colspan="6">Protocolo de capa física</td></tr>
</table>

Figura 7-59. La pila de protocolos H.323.

Para ver cómo encajan todos estos protocolos, considere el caso de una terminal de PC en una LAN (con un guardián) que llama a un teléfono remoto. Primero, la PC tiene que descubrir el guardián, por lo que difunde un paquete UDP de descubrimiento de guardián en el puerto 1718. Cuando el guardián responde, la PC aprende su dirección IP. Ahora la PC se registra con el guardián para lo cual le envía un mensaje RAS en un paquete UDP. Después de ser aceptada, la PC envía al guardián un mensaje de admisión RAS para solicitar ancho de banda. Sólo después de que se le ha otorgado el ancho de banda es cuando puede iniciar el establecimiento de la llamada. La idea de solicitar ancho de banda por adelantado es para permitir al guardián limitar el número de llamadas. Así puede evitar exceder el límite de solicitudes de la línea de salida, para ayudar a proveer la calidad de servicio necesaria.

Como observación adicional, el sistema telefónico hace lo mismo. Cuando usted levanta la bocina, se envía una señal a la oficina final local. Si ésta tiene suficiente capacidad para otra llamada, genera un tono de marcado. Si no, no se escuchará nada. En la actualidad, el sistema está tan sobredimensionado que el tono de marcado casi siempre es instantáneo; pero en los primeros días de la telefonía, a menudo tardaba unos cuantos segundos. Entonces, si alguna vez su nieto le pregunta: “¿Por qué hay tonos de marcado?”, ahora le podrá explicar. Excepto que, para entonces, tal vez los teléfonos ya no existan.

Ahora la PC establece una conexión TCP con el guardián para empezar a establecer la llamada. Aquí se utilizan los protocolos de la red telefónica existente que están orientados a conexión, por lo cual se necesita TCP. En contraste, el sistema telefónico no tiene nada parecido a RAS que permita a los teléfonos anunciar su presencia por lo que los diseñadores del H.323 tenían la libertad de usar UDP o TCP para RAS, y eligieron el UDP por su baja sobrecarga.

Ahora que tiene asignado el ancho de banda, la PC puede enviar un mensaje *SETUP* del protocolo Q.931 a través de la conexión TCP. Este mensaje especifica el número del teléfono al que se va a llamar (o la dirección IP y puerto, si se va a llamar a una computadora). El guardián responde con un mensaje

CALL PROCEEDING de Q.931 para confirmar la recepción correcta de la solicitud. A continuación, el guardián reenvía el mensaje *SETUP* a la puerta de enlace.

La puerta de enlace, que es mitad computadora y mitad conmutador telefónico, realiza entonces una llamada telefónica ordinaria al teléfono (ordinario) deseado. La oficina final a la que está conectado el teléfono hace sonar al teléfono al que se está llamando y también envía de vuelta un mensaje *ALERT* de Q.931 para indicar a la PC que hace la llamada que el teléfono ha empezado a sonar. Cuando la persona en el otro extremo levanta el teléfono, la oficina final envía de vuelta un mensaje *CONNECT* de Q.931 para indicar a la PC que tiene una conexión.

Una vez establecida la conexión, el guardián ya no está dentro del ciclo, aunque la puerta de enlace sí, desde luego. Los paquetes posteriores pasan por alto al guardián y van directamente a la dirección IP de la puerta de enlace. En este punto, sólo tenemos un simple tubo que conecta a las dos partes. Es sólo una conexión de capa física para desplazar bits, y nada más. Ninguno de los lados sabe nada acerca del otro.

Ahora se usa el protocolo H.245 para negociar los parámetros de la llamada. Utiliza el canal de control H.245, que siempre está abierto. Cada lado empieza por anunciar sus capacidades; por ejemplo, si puede manejar video (H.323 puede manejar video) o llamadas de conferencias, los codecs que soporta, etc. Una vez que cada lado conoce lo que el otro puede manejar se establecen dos canales de datos unidireccionales y se asigna un códec junto con otros parámetros a cada canal. Como cada lado puede tener distinto equipo, es totalmente posible que los códecs en el canal para enviar y el canal para recibir sean diferentes. Después de completar todas las negociaciones, puede empezar el flujo de datos mediante el uso de RTP. Se administra mediante RCTP, que desempeña su función para el control de la congestión. Si hay video presente, RTCP se encarga de la sincronización entre audio y video. En la figura 7-60 se muestran los diversos canales. Cuando alguna de las partes cuelga, se utiliza el canal de señalización de llamada de Q.931 para terminar la conexión después de haber completado la llamada para liberar los recursos que ya no se necesitan.

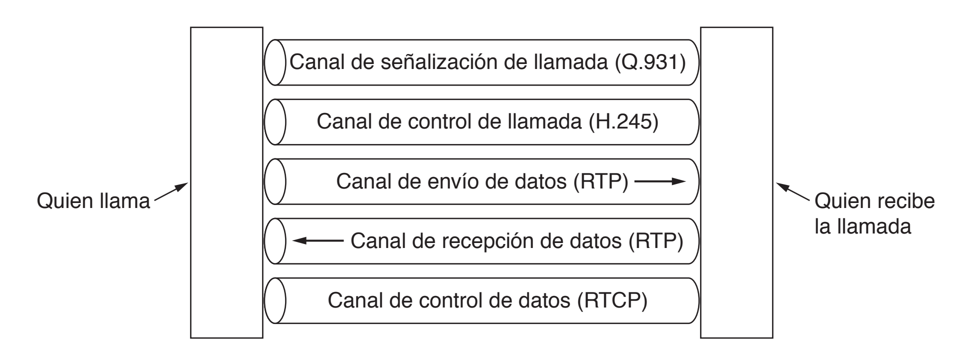


Figura 7-60. Canales lógicos entre el que hace la llamada y el que la recibe durante la llamada.

Al terminar la llamada, la PC que la inició se contacta de nuevo con el guardián mediante un mensaje RAS para liberar el ancho de banda que se le asignó. O también puede hacer otra llamada.

No hemos dicho nada sobre la calidad del servicio como parte de H.323, aun cuando dijimos que es una parte importante para convertir a las conferencias de tiempo real en un éxito. La razón es que la QoS queda fuera del alcance del H.323. Si la red subyacente es capaz de producir una conexión estable sin variación en el retardo desde la PC que hace la llamada hasta la puerta de enlace, la QoS en la llamada será buena; en caso contrario, no lo será. Sin embargo, en ninguna parte de la llamada en el lado del teléfono habrá variación en el retardo, debido a que es la forma en que se diseñó la red telefónica.

SIP: Protocolo de Inicio de Sesión

La ITU diseñó el H.323. Muchas personas en la comunidad de Internet lo vieron como un producto típico de compañía telefónica: grande, complejo e inflexible. En consecuencia, la IETF estableció un comité para diseñar una forma más simple y modular de llevar a cabo las llamadas de voz sobre IP. El resultado más importante a la fecha es **SIP** (**Protocolo de Inicio de Sesión**, del inglés *Session Initiation Protocol*). La versión más reciente se describe en el RFC 3261, el cual se escribió en el año 2002. Este protocolo describe cómo establecer llamadas telefónicas en Internet, conferencias de video y otras conexiones multimedia. A diferencia del H.323, que es una suite completa de protocolos, SIP es un solo módulo, pero se diseñó para funcionar bien con las aplicaciones existentes en Internet. Por ejemplo, define los números telefónicos como direcciones URL, por lo que las páginas web pueden contenerlos y para iniciar una llamada telefónica sólo hay que hacer clic en ellos (de la misma forma en que el esquema *mailto* permite hacer clic en un vínculo para iniciar un programa que envíe un mensaje de correo electrónico).

SIP puede establecer sesiones entre dos partes (llamadas telefónicas ordinarias), sesiones multipartitas (en donde todos pueden escuchar y hablar) y sesiones multidifusión (un emisor, muchos receptores). Las sesiones pueden contener audio, video o datos; este último caso es útil para los juegos multijugador en tiempo real, por ejemplo. SIP sólo se encarga de establecer, administrar y terminar las sesiones. Otros protocolos, como RTP/RTCP, se utilizan también para el transporte de datos. SIP es un protocolo de capa de aplicación y puede operar sobre UDP o TCP, según se requiera.

SIP soporta una variedad de servicios, incluyendo localizar al que recibe la llamada (que tal vez no se encuentre en la máquina en su hogar) y determinar sus capacidades, así como manejar los mecanismos de establecimiento y terminación de las llamadas. En el caso más simple, SIP establece una sesión desde la computadora del que hace la llamada hasta la computadora del que la recibe, por lo que examinaremos primero ese caso.

En SIP, los números telefónicos se representan como URL mediante el uso del esquema *sip*; por ejemplo, *sip:ilse@cs.university.edu* para un usuario llamado Ilse en el host especificado por el nombre DNS *cs.university.edu*. Los URL de SIP también pueden contener direcciones IPv4, direcciones IPv6 o números telefónicos reales.

El protocolo SIP es un protocolo basado en texto, modelado con base en la HTTP. Una parte envía un mensaje en texto ASCII que consiste en el nombre de un método en la primera línea, seguida de líneas adicionales que contienen encabezados para el paso de parámetros. Muchos de los encabezados se toman de MIME para permitir que SIP interfuncione con las aplicaciones existentes en Internet. En la figura 7.61 se listan los seis métodos definidos por la especificación básica.

Para establecer una sesión, el que hace la llamada crea una conexión TCP con el que la recibe y envía un mensaje *INVITE* a través de ella, o envía el mensaje *INVITE* en un paquete UDP. En ambos casos, los

Método	Descripción
INVITE	Solicitar el inicio de una sesión.
ACK	Confirmar que se inició una sesión.
BYE	Solicitar la terminación de una sesión.
OPTIONS	Consultar a un host sobre sus capacidades.
CANCEL	Cancelar una solicitud pendiente.
REGISTER	Informar a un servidor de redirección sobre la ubicación actual del usuario.

Figura 7-61. Métodos de SIP.

encabezados de la segunda línea y de las líneas posteriores describen la estructura del cuerpo del mensaje, el cual contiene las capacidades del que hace la llamada, los tipos de medios y los formatos. Si el que recibe la llamada la acepta, responde con un código de respuesta tipo HTTP (un número de tres dígitos que utiliza los grupos de la figura 7-38, 200 para la aceptación). Después de la línea de código de respuesta, el que recibe la llamada también puede proveer información sobre sus capacidades, tipos de medios y formatos.

La conexión se realiza mediante el uso de un acuerdo de tres vías, por lo que el que llama responde con un mensaje *ACK* para finalizar el protocolo y confirmar la recepción del mensaje 200.

Cualquiera de las partes puede solicitar la terminación de una sesión mediante el envío de un mensaje con el método *BYE*. Cuando el otro lado confirma la recepción de ese mensaje, se termina la sesión.

El método *OPTIONS* se utiliza para consultar a una máquina sobre sus propias capacidades. Se utiliza por lo general antes de iniciar una sesión, para averiguar si esa máquina es capaz de realizar llamadas de voz sobre IP o cualquier tipo de sesión que se contemple.

El método *REGISTER* se relaciona con la habilidad de SIP de rastrear y conectarse con un usuario que esté alejado de su base. Este mensaje se envía a un servidor de ubicación SIP que mantiene el registro de la posición de cada quien. Más tarde podemos consultar a ese servidor para encontrar la ubicación actual del usuario. En la figura 7-62 se muestra la operación de redirección. Aquí, el que llama envía el mensaje *INVITE* a un servidor proxy para ocultar la posible redirección. Después el proxy busca la ubicación del usuario y envía el mensaje *INVITE* ahí. Luego actúa como retransmisor para los mensajes posteriores en el acuerdo de tres vías. Los mensajes *LOOKUP* y *REPLY* no forman parte del SIP; se puede usar cualquier protocolo conveniente, dependiendo del tipo de servidor y de ubicación que se utilice.

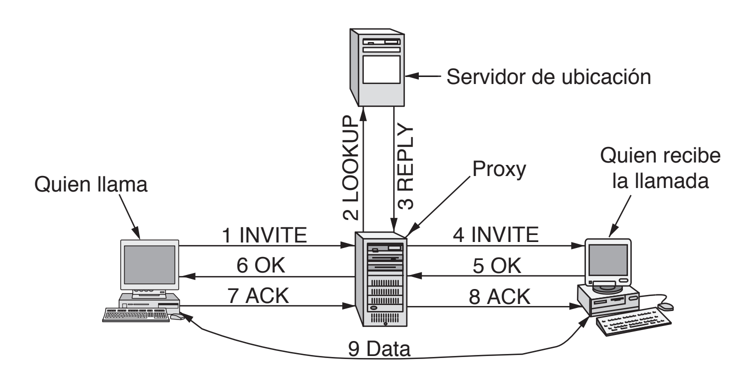


Figura 7-62. Uso de un servidor proxy y redirección mediante SIP.

El SIP tiene varias características más que no describiremos aquí, entre las cuales están: llamada en espera, verificación de llamadas, cifrado y autentificación. También tiene la habilidad de realizar llamadas de una computadora a un teléfono ordinario, si está disponible una puerta de enlace adecuada entre Internet y el sistema telefónico.

Comparación entre H.323 y SIP

Tanto H.323 como SIP permiten llamadas de dos partes y multipartitas mediante el uso de computadoras y teléfonos como puntos terminales. Ambos soportan la negociación de parámetros, el cifrado y los protocolos RTP/RTCP. En la figura 7-63 se muestra un resumen de sus similitudes y diferencias.

Aunque los conjuntos de características son similares, los dos protocolos son muy distintos en cuanto a su filosofía. El H.323 es un estándar típico pesado de la industria telefónica, que especifica la pila completa de protocolos y define con precisión lo que se permite y se prohíbe. Esta metodología conduce

Característica	H.323	SIP
Diseñado por	ITU	IETF
Compatibilidad con PSTN	Sí	En gran parte
Compatibilidad con Internet	Sí, con el tiempo	Sí
Arquitectura	Monolítica	Modular
Grado de complejidad	Pila de protocolos completa	SIP sólo maneja el establecimiento
Negociación de parámetros	Sí	Sí
Señalización de llamadas	Q.931 sobre TCP	SIP sobre TCP o UDP
Formato de mensajes	Binario	ASCII
Transporte de medios	RTP/RTCP	RTP/RTCP
Llamadas multipartitas	Sí	Sí
Conferencias multimedia	Sí	No
Direccionamiento	URL o número telefónico	URL
Terminación de llamada	Explícita o liberación TCP	Explícita o expiración de temporizador
Mensajería instantánea	No	Sí
Cifrado	Sí	Sí
Tamaño de estándares	1,400 páginas	250 páginas
Implementación	Grande y compleja	Moderada, pero hay problemas
Estado	Extendido, esp. video	Alternativo, esp. Voz

Figura 7-63. Comparación entre H.323 y SIP.

a protocolos muy bien definidos en cada capa, lo cual facilita la tarea de la interoperabilidad. El precio a pagar es un estándar extenso, complejo y rígido, difícil de adaptar a las futuras aplicaciones.

En contraste, el SIP es un protocolo típico de Internet que funciona mediante el intercambio de líneas cortas de texto ASCII. Es un módulo ligero que interfunciona bien con otros protocolos de Internet, pero no tan bien con los protocolos de señalización del sistema telefónico existente. Como el modelo de voz sobre IP de la IETF es muy modular, es flexible y se adapta con facilidad a las nuevas aplicaciones. La desventaja es que ha sufrido de problemas continuos de interoperabilidad, a medida que las personas tratan de interpretar lo que significa el estándar.

7.5 ENTREGA DE CONTENIDO

Al principio Internet sólo servía como canal de comunicación, al igual que la red telefónica. En un principio los académicos se comunicaban con las máquinas remotas, para iniciar sesión a través de la red y realizar tareas. Las personas han usado el correo electrónico para comunicarse entre sí durante mucho tiempo, y ahora usan también video y voz sobre IP. Sin embargo, como la web creció, Internet trata ahora más sobre contenido que comunicación. Muchas personas usan la web para buscar información y hay una enorme actividad de compartición de archivos de igual a igual impulsada por el acceso a las películas, la

música y los programas. El cambio al contenido ha sido tan pronunciado que la mayor parte del ancho de banda de Internet se utiliza ahora para transmitir videos almacenados.

Como la tarea de distribuir contenido es distinta de la de comunicación, los requerimientos impuestos sobre la red son distintos. Por ejemplo, si Sally desea hablar con Jitu, puede realizar una llamada de voz sobre IP a su teléfono móvil. La comunicación debe ser con una computadora específica; no servirá de nada llamar a la computadora de Paul. Pero si Jitu desea ver el partido de cricket más reciente de su equipo, le basta con ver video de flujo continuo desde cualquier computadora que pueda proporcionar el servicio. No le importa si la computadora es de Sally o de Paul o, lo que es más probable, un servidor desconocido en Internet. Es decir, la ubicación no importa para el contenido, excepto en cuanto a que afecta el desempeño (y la legalidad).

La otra diferencia es que ciertos sitios web que proveen contenido se han vuelto en extremo populares. YouTube es uno de los principales ejemplos. Permite a los usuarios compartir videos de su creación sobre cualquier tema concebible. Muchas personas desean hacer esto. El resto de nosotros queremos observar. Con todos estos videos que ocupan grandes cantidades de ancho de banda, se estima que YouTube es responsable de 10% del tráfico de Internet en la actualidad.

Ningún servidor es lo bastante poderoso o confiable como para manejar por sí solo un nivel tan asombroso de demanda. En vez de ello, YouTube y otros grandes proveedores de contenido crean sus propias redes de distribución de contenido. Estas redes usan centros de datos esparcidos en todo el mundo para servir contenido a un número extremadamente grande de clientes con buenos niveles de desempeño y confiabilidad.

Las técnicas que se utilizan para la distribución de contenido se han desarrollado a través del tiempo. Cuando la web empezó a crecer, su popularidad fue casi su ruina. Las demandas crecientes de contenido sobrecargaban con frecuencia a los servidores y las redes. Muchas personas empezaron a usar el apodo "World Wide Wait" en inglés, que se traduce como "espera a nivel mundial" para la WWW.

En respuesta a la demanda de los consumidores, el núcleo de Internet se aprovisionó de grandes cantidades de ancho de banda y se extendió una conectividad de banda ancha más rápida en el extremo de la red. Este ancho de banda fue la clave para mejorar el desempeño, pero sólo es una parte de la solución. Para reducir los retardos interminables, los investigadores también desarrollaron distintas arquitecturas para usar el ancho de banda para distribuir contenido.

Una de esas arquitecturas es la **CDN** (**Red de Distribución de Contenido**, del inglés *Content Distribution Network*). En ella, un proveedor establece una colección distribuida de máquinas en ubicaciones dentro de Internet y las utiliza para servir contenido a los clientes. Ésta es la elección de los grandes participantes. Una arquitectura alternativa es la red **P2P** (**Igual a Igual**, del inglés *Peer-to-Peer*). En ella, una colección de computadoras reservan sus recursos para servir contenido unas a otras, sin servidores aprovisionados por separado y sin un punto central de control. Esta idea ha capturado la imaginación de las personas ya que, al actuar en conjunto, muchos pequeños participantes pueden formar un enorme grupo.

En esta sección analizaremos el problema de distribuir contenido en Internet y algunas de las soluciones que se utilizan en la práctica. Después de discutir con brevedad sobre la popularidad del contenido y el tráfico en Internet, describiremos cómo crear servidores web poderosos y usar la caché para mejorar el desempeño para los clientes web. Entonces analizaremos las dos principales arquitecturas para distribuir contenido: las redes CDN y P2P. Sus diseños y propiedades son bastante diferentes, como veremos.

7.5.1 Contenido y tráfico de Internet

Para diseñar y gestionar redes que funcionen bien, necesitamos comprender el tráfico que deben transportar. Por ejemplo, con el cambio del tráfico a contenido, los servidores han migrado de las oficinas empresariales a centros de datos en Internet que proveen grandes cantidades de máquinas con una

excelente conectividad de red. Para ejecutar incluso un servidor pequeño en la actualidad, es más fácil y económico rentar un servidor virtual hospedado en un centro de datos de Internet que operar una máquina real en un hogar u oficina con conectividad de banda ancha a Internet.

Por fortuna, sólo hay dos factores sobre el tráfico de Internet que es esencial conocer. El primer factor es que cambia con rapidez, no sólo en cuanto a los detalles sino en su composición en general. Antes de 1994, la mayoría del tráfico constaba de transferencias de archivos por FTP tradicional (para desplazar programas y conjuntos de datos entre computadoras) y de correo electrónico. Después llegó la web y creció en forma exponencial. El tráfico web dejó atrás al tráfico de FTP y de correo electrónico mucho antes de la burbuja punto com del año 2000. A partir de este año, empezó a hacerse notar la compartición de archivos P2P, primero para la música y después para las películas. Para 2003, la mayoría del tráfico de Internet era tráfico P2P, que dejó atrás al tráfico web en una nube de polvo. A finales de la década de 2000, el video transmitido en flujo continuo mediante los métodos de distribución de contenido en sitios como YouTube empezó a exceder al tráfico P2P. Para 2014, Cisco pronostica que 90% de todo el tráfico en Internet será de video, en una forma o en otra (Cisco, 2010).

No siempre es el volumen de tráfico lo que importa. Por ejemplo, mientras que el tráfico de voz sobre IP se expandió en forma explosiva incluso antes de que Skype iniciara en 2003, siempre será un pequeño punto en el radar debido a que los requerimientos de ancho de banda del audio son dos veces menores que los del video. Sin embargo, el tráfico de voz sobre IP presiona a la red de otras formas, ya que es sensible a la latencia. Como otro ejemplo, las redes sociales en línea han crecido en forma dramática desde que Facebook inició sus operaciones en 2004. En 2010, por primera vez, Facebook llegaba a más usuarios en la web por día que Google. Aún si hacemos a un lado el tráfico (y sin duda es mucho tráfico del que hablamos), las redes sociales en línea son importantes debido a que están cambiando la forma en que las personas interactúan a través de Internet.

El punto al que queremos llegar es que los desplazamientos sísmicos en el tráfico de Internet ocurren con rapidez, y con cierta regularidad. ¿Qué seguirá ahora? Por favor, regrese cuando esté lista la sexta edición de este libro y le haremos saber.

El segundo factor esencial sobre el tráfico de Internet es que es altamente sesgado. Muchas propiedades con las que estamos familiarizados se agrupan alrededor de un promedio. Por ejemplo, la mayoría de los adultos están cerca de la altura promedio. Hay algunas personas altas y otras bajas, pero pocas son muy altas o muy bajas. Para estos tipos de propiedades es posible diseñar teniendo en cuenta un rango que no sea muy grande, pero que capture a la mayoría de la población.

El tráfico de Internet no es así. Durante mucho tiempo se ha sabido que existe una pequeña cantidad de sitios web con tráfico masivo y una gran cantidad de sitios web con tráfico mucho menor. Esta característica se ha vuelto parte del lenguaje de las redes. Las primeras publicaciones hablaban sobre el tráfico en términos de trenes de paquetes, con base en la idea de que pronto habría trenes exprés con una gran cantidad de paquetes viajando por un enlace (Jain y Routhier, 1986). Esto se formalizó como la noción de la **autosimilitud**, que para nuestros fines podemos considerar como el tráfico de red que exhibe muchos vacíos cortos y largos, aun cuando se observe desde distintas escalas de tiempo (Leland y colaboradores, 1994). El trabajo posterior hablaba de los flujos extensos de tráfico como si fueran **elefantes** y de los flujos cortos de tráfico como **ratones**. La idea es que sólo haya unos cuantos elefantes y muchos ratones, pero los elefantes son importantes debido a que son muy grandes.

Ahora regresemos al contenido web. Aquí es evidente el mismo tipo de sesgo. La experiencia con las tiendas de renta de video, bibliotecas públicas y otras organizaciones similares muestra que no todos los elementos tienen la misma popularidad. Hablando en sentido experimental, cuando hay N películas disponibles, la fracción de todas las solicitudes por la k-ésima más popular es de aproximadamente C/k. Aquí, C se calcula para normalizar la suma a 1, es decir,

$$C = 1/(1 + 1/2 + 1/3 + 1/4 + 1/5 + \cdots + 1/N)$$

Así, la película más popular es siete veces más popular que la película número siete. Este resultado se conoce como la **ley de Zipf** (Zipf, 1949). Se nombró en honor de George Zipf, un profesor de lingüística en la Universidad de Harvard, quien notó que la frecuencia del uso de una palabra en un gran conjunto de texto es inversamente proporcional a su rango. Por ejemplo, la 40va palabra más común se usa dos veces más que la 80va palabra más común, y tres veces más que la 120va palabra más común.

En la figura 7-64(a) se muestra una distribución de Zipf, la cual captura la noción de que hay una pequeña cantidad de elementos populares y una gran cantidad de no populares. Para reconocer las distribuciones de esta forma, es conveniente trazar los datos en una escala logarítmica en ambos ejes, como se muestra en la figura 7-64(b). El resultado deberá ser una línea recta.

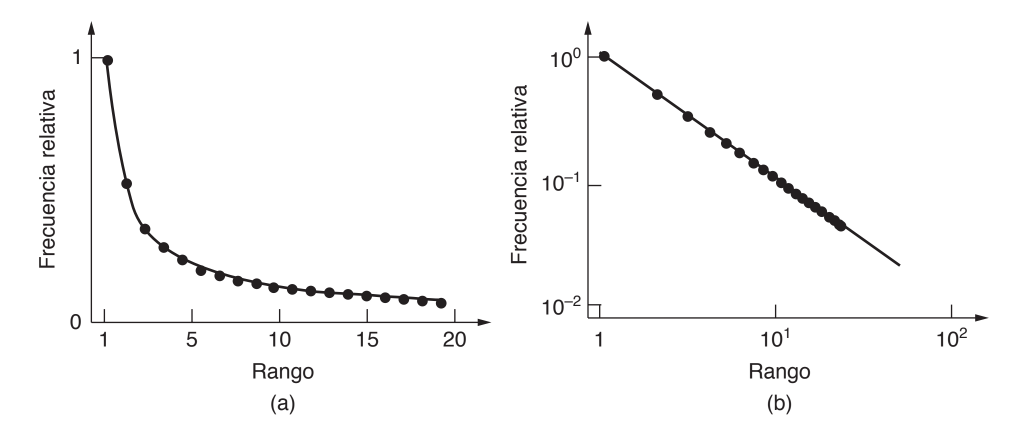


Figura 7-64. Distribución Zipf (a) En una escala lineal. (b) En una escala log-log.

Cuando las personas vieron la popularidad de las páginas web, también dio la casualidad de que seguían la ley de Zipf en forma aproximada (Breslau y colaboradores, 1999). Una distribución de Zipf es un ejemplo en una familia de distribuciones conocidas como **leyes de potencia**. Estas leyes son evidentes en muchos fenómenos humanos, como la distribución de las poblaciones de ciudades y de salud. Tienen la misma propensión a describir unos cuantos participantes grandes y muchos participantes pequeños, y también aparecen como línea recta en un gráfico log-log. Pronto se descubrió que la topología de Internet podía describirse de manera aproximada mediante las leyes de potencia (Faloutsos y colaboradores, 1999). Después los investigadores empezaron a trazar todas las propiedades imaginables de Internet en una escala logarítmica, observaron una línea recta y gritaron: "¡Ley de potencia!".

Sin embargo, lo que importa más que una línea recta en un gráfico log-log es lo que significan estas distribuciones para el diseño y uso de las redes. Dadas las diversas formas de contenido que tienen distribuciones de la ley de Zipf o de potencia, parece ser fundamental que los sitios web en Internet sean similares a Zipf en cuanto a popularidad. Esto a su vez significa que un sitio promedio no es una representación útil. Los sitios se describen mejor como populares o no populares. Ambos tipos de sitios son importantes. Sin duda los sitios populares importan, ya que unos cuantos sitios populares pueden ser responsables de la mayoría del tráfico en Internet. Tal vez lo sorprendente sea que los sitios no populares también pueden importar. Esto se debe a que la cantidad total de tráfico dirigido a los sitios no populares puede llegar a constituir en conjunto una gran parte del tráfico en general. La razón es que hay muchos de estos sitios no populares. La noción de que muchas elecciones no populares pueden importar en conjunto se ha popularizado a través de libros como *The Long Tail* (Anderson, 2008a).

Son comunes las curvas que muestran decadencia, como en la figura 7-64(a), pero no todas son iguales. En especial, las situaciones en las que la tasa de decadencia es proporcional a la cantidad de

material restante (como con los átomos radiactivos inestables) exhiben una **decadencia exponencial**, la cual decrece mucho más rápido que la Ley de Zipf. El número de elementos (digamos, átomos) restantes después del tiempo t se expresa por lo general como $e^{-t/\alpha}$, en donde la constante α determina qué tan rápida es la decadencia. La diferencia entre la decadencia exponencial y la Ley de Zipf es que con la primera, es seguro ignorar el final de la cola, pero con la segunda el peso total de la cola es considerable y no se puede ignorar.

Para trabajar de manera efectiva en este mundo sesgado, debemos ser capaces de construir ambos tipos de sitios web. Los sitios no populares son fáciles de manejar. Mediante el uso del DNS, muchos sitios distintos en realidad apuntan a la misma computadora en Internet que opera todos esos sitios. Por otro lado, los sitios populares son difíciles de manejar. No hay una sola computadora que se acerque siquiera a la potencia requerida, y si la hubiera, el hecho de usar una sola computadora significaría que el sitio estaría inaccesible a millones de usuarios si ésta fallara. Para manejar estos sitios, debemos construir sistemas de distribución de contenido. Empezaremos con esta tarea en la siguiente sección.

7.5.2 Granjas de servidores y proxies web

Los diseños web que hemos visto hasta ahora tienen una sola máquina servidor que se comunica con varias máquinas cliente. Para crear grandes sitios web que tengan un buen desempeño, podemos agilizar el procesamiento, ya sea en el lado servidor o en el lado cliente. En el lado servidor, se pueden crear servidores web más poderosos con una granja de servidores, en donde un clúster de computadoras actúa como un solo servidor. En el lado cliente, se puede lograr un mejor desempeño con mejores técnicas de almacenamiento en caché. En especial, las cachés de proxy proveen una caché grande compartida para un grupo de clientes.

Describiremos cada una de estas técnicas por separado. Sin embargo, cabe mencionar que ninguna técnica es suficiente para crear los sitios web más grandes. Esos sitios populares requieren los métodos de distribución de contenido que describiremos en las siguientes secciones, los cuales combinan computadoras en muchas ubicaciones distintas.

Granjas de servidores

Sin importar cuánto ancho de banda tenga una máquina, sólo puede atender todas las solicitudes web que le lleguen antes de que la carga sea demasiado grande. La solución en este caso es usar más de una computadora para crear un servidor web. Esto nos lleva al modelo de **granja de servidores** de la figura 7-65.

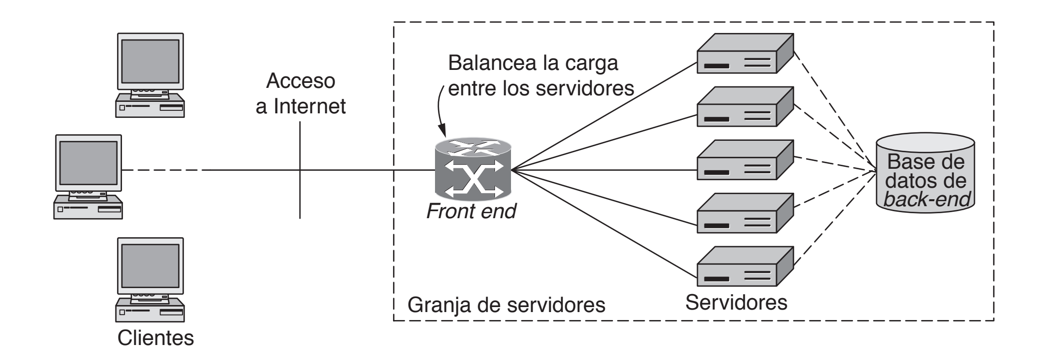


Figura 7-65. Una granja de servidores.

La dificultad con este modelo aparentemente sencillo es que el conjunto de computadoras que conforman la granja de servidores debe parecer un solo sitio web lógico para los clientes. Si no es así, entonces sólo hemos establecido distintos sitios web que operan en paralelo.

Hay varias soluciones posibles para hacer que el conjunto de servidores parezca ser un solo sitio web. Todas las soluciones suponen que cualquiera de los servidores puede atender una solicitud de cualquier cliente. Para ello, cada servidor debe tener una copia del sitio web. Los servidores se muestran como conectados a una base de datos común de *back-end*, mediante una línea punteada para este fin.

Una solución es usar DNS para dispersar las solicitudes entre los servidores en la granja. Cuando se hace una solicitud de DNS para el URL del sitio web, el servidor DNS devuelve una lista rotativa de las direcciones IP de los servidores. Cada cliente prueba una dirección IP, por lo general la primera de la lista. El efecto es que los distintos clientes se contacten con diferentes servidores para acceder al mismo sitio web, como se tenía planeado. El método del DNS está en el núcleo de las redes CDN, por lo que volveremos a analizarlo más tarde en esta sección.

Las otras soluciones se basan en un **front-end** que esparce las solicitudes entrantes a través de la reserva de servidores en la granja de servidores. Esto ocurre incluso cuando el cliente se pone en contacto con la granja de servidores mediante el uso de una sola dirección IP de destino. Por lo general el *front-end* es un switch de capa de enlace o un enrutador IP; es decir, un dispositivo que maneja tramas o paquetes. Todas las soluciones se basan en que el *front-end* (o los servidores) hurgue en los encabezados de capa de red, transporte o aplicación y los utilice de maneras no estándar. Una solicitud web y una respuesta se transportan como una conexión TCP. Para funcionar de manera correcta, el *front-end* debe distribuir todos los paquetes de una solicitud al mismo servidor.

Un diseño simple es que el *front-end* difunda todas las solicitudes entrantes a todos los servidores. Cada servidor responde sólo a una fracción de las solicitudes mediante un acuerdo previo. Por ejemplo, 16 servidores podrían ver la dirección IP de origen y responder a la solicitud sólo si los últimos 4 bits de la dirección IP de origen coinciden con sus selectores configurados. Otros paquetes se descartan. Aunque esto es un desperdicio de ancho de banda entrante, con frecuencia las respuestas son mucho más largas que la solicitud, por lo que no es tan ineficiente como suena.

En un diseño más general, el *front-end* puede inspeccionar los encabezados IP, TCP y HTTP de los paquetes y asociarlos de manera arbitraria a un servidor. A esta asociación se le denomina política de **balanceo de carga**, ya que el objetivo es balancear la carga de trabajo entre los servidores. Esta política puede ser simple o compleja. Una política simple podría ser la de usar los servidores uno después del otro por turnos, o por turnos rotatorios (*round-robin*). Con esta metodología el *front-end* debe recordar la asociación para cada solicitud, de manera que los paquetes posteriores que sean parte de la misma solicitud se envíen al mismo servidor. Además, para que el sitio sea más confiable que con un solo servidor, el *front-end* debe detectar cuando fallen los servidores y dejar de enviarles solicitudes.

Al igual que NAT, este diseño general es arriesgado, o cuando menos frágil, en cuanto a que acabamos de crear un dispositivo que viola el principio más básico de los protocolos por capas: cada capa debe usar su propio encabezado para fines de control y no puede inspeccionar o usar la información de la carga útil para ningún propósito. Pero las personas diseñan dichos sistemas de todas formas, y cuando después fallan debido a cambios en las capas superiores, tienden a sorprenderse. En este caso, el *front-end* es un switch o un enrutador, pero puede tomar una acción con base en la información de la capa de transporte o de una capa superior. A dicha caja se le conoce como **middlebox** debido a que se interpone en medio de la trayectoria de red con la cual no tiene nada que ver, de acuerdo con la pila de protocolos. En este caso, el *front-end* se considera mejor como una parte interna de una granja de servidores que termina todas las capas, hasta la capa de aplicación (y por ende, puede usar la información de todos los encabezados de esas capas).

Sin embargo, al igual que con NAT, este diseño es útil en la práctica. La razón de analizar los encabezados TCP es que es posible hacer un mejor trabajo de balanceo de carga que con la información de IP por

sí sola. Por ejemplo, una dirección IP puede representar toda una empresa y realizar muchas solicitudes. Sólo mediante un análisis de la información de TCP o de capas superiores se podrán asociar estas solicitudes a distintos servidores.

La razón de analizar los encabezados HTTP es algo distinta. Muchas interacciones web acceden a bases de datos y las actualizan, como cuando un cliente busca su compra más reciente. El servidor que cubra esta solicitud tendrá que consultar a la base de datos de *back-end*. Es conveniente dirigir las solicitudes posteriores que provengan del mismo usuario al mismo servidor, ya que éste contendrá de antemano información en su caché sobre ese usuario. La manera más simple de hacer que ocurra esto es mediante el uso de cookies de web (o de otro tipo de información para distinguir al usuario) e inspeccionar los encabezados de HTTP para encontrar esas cookies.

Como observación final, aunque hemos descrito este diseño para los sitios web, podemos construir una granja para otros tipos de servidores también. Un ejemplo son los servidores de medios de flujo continuo sobre UDP. El único cambio que se requiere es que el *front-end* pueda balancear la carga de estas solicitudes (que tendrán distintos campos en los encabezados de los protocolos a los de las solicitudes web).

Proxies web

Las solicitudes y respuestas web se envían mediante HTTP. En la sección 7.3 describimos cómo los navegadores pueden almacenar las respuestas en la caché y después reutilizarlas para responder a las solicitudes posteriores. El navegador utiliza varios campos de encabezado y reglas para determinar si una copia en caché de una página web sigue siendo actual. No repetiremos aquí este material.

El almacenamiento en caché mejora el desempeño al acortar el tiempo de respuesta y reducir la carga de la red. Si el navegador puede determinar que una página en caché es actual por sí solo, se puede obtener la página de la caché de inmediato, sin ningún tráfico en la red. Incluso aunque el navegador pida al servidor la confirmación de que la página aún es actual, se acorta el tiempo de respuesta y se reduce la carga de la red, en especial para las páginas extensas, ya que sólo hay que enviar un pequeño mensaje.

Sin embargo, lo mejor que puede hacer el navegador es almacenar en caché todas las páginas web que el usuario haya visitado antes. De nuestro análisis sobre la popularidad, tal vez recuerde que así como hay unas cuantas páginas populares que las personas visitan repetidas veces, también hay muchas, pero muchas páginas no populares. En la práctica, esto limita la efectividad del almacenamiento en caché del navegador, ya que hay una gran cantidad de páginas que un usuario dado visita sólo una vez. Estas páginas siempre se tienen que obtener del servidor.

Una estrategia para aumentar la efectividad de las cachés es compartir la caché entre varios usuarios. De esta forma, una página que ya se haya obtenido para un usuario se puede devolver a otro usuario cuando éste haga la misma solicitud. Si el navegador no almacenara páginas en la caché, ambos usuarios tendrían que obtener la página del servidor. Desde luego que esta compartición no se puede realizar para el tráfico cifrado, las páginas que requieren autentificación y las páginas que devuelven los programas y que no se pueden almacenar en caché (como los precios actuales en el mercado de valores). En especial, las páginas dinámicas creadas por los programas son un caso creciente para el cual el almacenamiento en caché no es efectivo. Sin embargo, hay muchas páginas web que son visibles para muchos usuarios y se ven igual, sin importar qué usuario haga la solicitud (como las imágenes).

Un **proxy web** se utiliza para compartir una caché entre varios usuarios. Un proxy es un agente que actúa en beneficio de alguien más, como el usuario. Hay muchos tipos de proxies. Por ejemplo, un proxy ARP responde a las solicitudes ARP en beneficio de un usuario que está en otra parte (y no puede responder por sí mismo). Una proxy web obtiene las solicitudes web en beneficio de sus usuarios. Por lo general provee el almacenamiento en caché de las respuestas web, y como se comparte entre los usuarios, tiene una caché mucho más grande que la de un navegador.

Cuando se utiliza un proxy, la configuración típica es que una organización opere un proxy web para todos sus usuarios. La organización podría ser una empresa o un ISP. Ambos buscan beneficiarse al agilizar las solicitudes web para sus usuarios y reducir sus necesidades de ancho de banda. Aunque los precios fijos, independientemente del uso, son comunes para los usuarios finales, a la mayoría de las empresas y proveedores ISP se les cobra de acuerdo con el ancho de banda que utilizan.

Esta configuración se muestra en la figura 7-66. Para usar el proxy, cada navegador se configura para crear solicitudes de página al proxy, en vez de hacerlo al servidor real de la página. Si el proxy tiene la página, la devuelve de inmediato. Si no la tiene, obtiene la página del servidor, la agrega a la caché para un uso futuro y la devuelve al cliente que la solicitó.

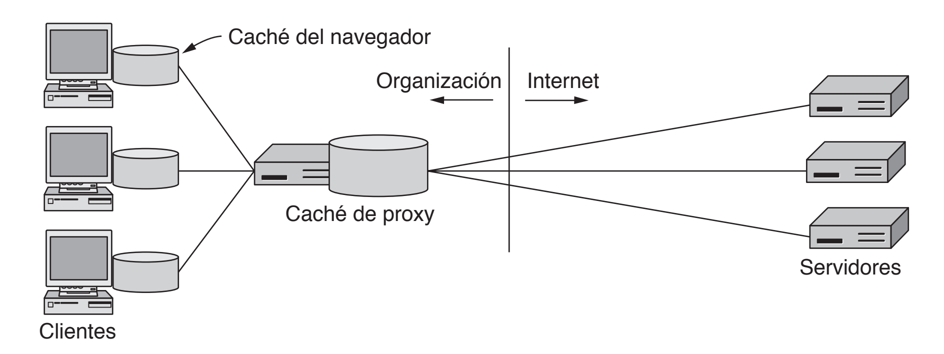


Figura 7-66. Una caché de proxy entre navegadores y servidores web.

Así como enviar solicitudes web al proxy en vez de hacerlo al servidor real, los clientes realizan su propio almacenamiento en caché mediante la caché de su navegador. El proxy sólo se consulta después de que el navegador haya tratado de satisfacer la solicitud de su propia caché. Es decir, el proxy provee un segundo nivel de almacenamiento en caché.

Se pueden agregar más proxies para proveer niveles adicionales de almacenamiento en caché. Cada proxy (o navegador) realiza solicitudes a través de su **proxy ascendente**. Cada proxy ascendente almacena páginas en la caché para los **proxies descendentes** (o navegadores). Por lo tanto, es posible que los navegadores en una empresa utilicen un proxy de la empresa, el cual usa un proxy del ISP, el cual se contacta de manera directa con los servidores web. Sin embargo, el único nivel de almacenamiento en caché de proxy que hemos mostrado en la figura 7-66 es a menudo suficiente para obtener la mayoría de los beneficios potenciales, en la práctica. De nuevo, el problema es la larga cola de la popularidad. Los estudios del tráfico web han mostrado que el almacenamiento en una caché compartida es benéfico hasta que el número de usuarios se aproxima al tamaño de una empresa pequeña (por decir, de 100 personas). A medida que el número de personas aumenta, los beneficios de compartir una caché se vuelven marginales gracias a las solicitudes no populares que no se pueden almacenar en caché debido a la carencia de espacio de almacenamiento (Wolman y colaboradores, 1999).

Los proxies web ofrecen beneficios adicionales que con frecuencia son un factor en la decisión para implementarlos. Uno de esos beneficios es filtrar el contenido. El administrador puede configurar el proxy para poner sitios en la lista negra, o filtrar de alguna otra forma las solicitudes que realiza. Por ejemplo, muchos administradores se disgustan con los empleados que observan videos en YouTube (o peor aún, pornografía) dentro de sus horas de trabajo y establecen sus filtros de manera acorde. Otro beneficio de tener proxies es la privacidad o el anonimato, cuando el proxy protege la identidad del usuario del servidor.

7.5.3 Redes de entrega de contenido

Las granjas de servidores y los proxies web ayudan a construir sitios grandes y mejorar el desempeño web, pero no son suficientes para los sitios web verdaderamente populares que deben servir contenido a una escala global. Para estos sitios se necesita un enfoque distinto.

Las redes **CDNs** (**Redes de Entrega de Contenido**, del inglés *Content Delivery Networks*) ponen de cabeza la idea del almacenamiento en caché de la web tradicional. En vez de hacer que los clientes busquen una copia de la página solicitada en una caché cercana, es el proveedor quien coloca una copia de la página en un conjunto de nodos en distintas ubicaciones, e indica al cliente que utilice un nodo cercano como el servidor.

En la figura 7-67 se muestra un ejemplo de la trayectoria que siguen los datos cuando se distribuyen mediante una red CDN. Es un árbol. El servidor de origen en la CDN distribuye una copia del contenido a otros nodos en la CDN, en Sidney, Boston y Amsterdam, en este ejemplo. Esto se muestra con líneas punteadas. Después, los clientes obtienen páginas del nodo más cercano en la CDN. Esto se muestra con líneas continuas. De esta manera, los clientes en Sydney obtienen la copia de la página almacenada en Sydney; no obtienen la página del servidor de origen, el cual puede estar en Europa.

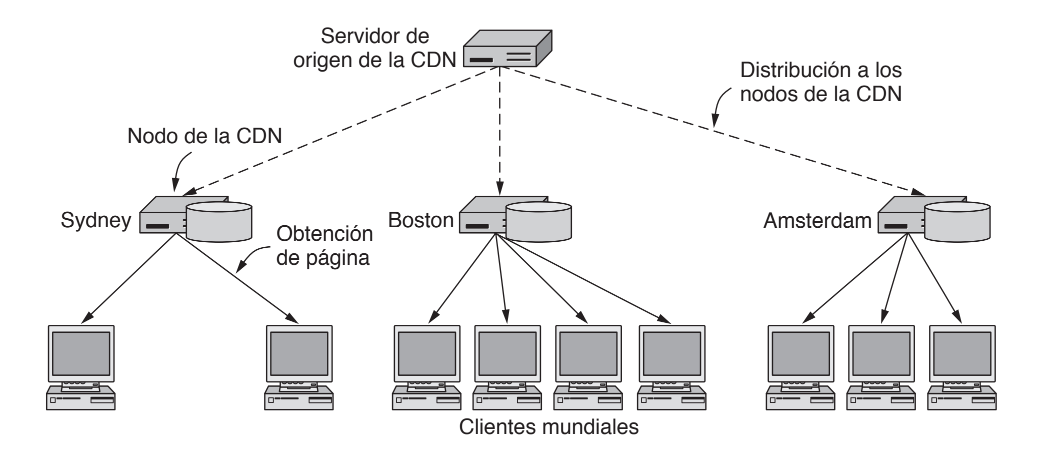


Figura 7-67. Árbol de distribución de una CDN.

Hay tres virtudes en cuanto al uso de una estructura de árbol. En primer lugar, la distribución de contenido se puede escalar hasta la cantidad de clientes que sean necesarios mediante el uso de más nodos en la CDN, y más niveles en el árbol cuando la distribución entre los nodos de la CDN se convierta en el "cuello de botella". No importa cuántos clientes haya, la estructura del árbol siempre es eficiente. El servidor de origen no se sobrecarga debido a que se comunica con todos los clientes por medio del árbol de nodos de la CDN; no tiene que responder a cada solicitud de una página por su cuenta. En segundo lugar, cada cliente obtiene un buen desempeño al obtener páginas de un servidor cercano en vez de uno distante. Esto se debe a que el tiempo de ida y vuelta requerido para establecer una conexión es más corto. El inicio lento de TCP se eleva con más rapidez debido al tiempo de ida y vuelta más corto; además es menos probable que la trayectoria de red más corta pase a través de regiones de congestión en Internet. Finalmente, la carga total que se coloca en la red también se mantiene en un valor mínimo. Si los nodos de la CDN están bien colocados, el tráfico para una página dada debe pasar a través de cada parte de la red sólo una vez. Esto es importante, ya que alguien debe pagar por el ancho de banda de la red en un momento dado.

La idea de usar un árbol de distribución es simple y directa. Lo que es menos simple es la forma de organizar los clientes para usar este árbol. Por ejemplo, tal vez los servidores de proxy podrían proveer una solución. Analizando la figura 7-67, si cada cliente se configurara para usar el nodo de la CDN de Sydney, Boston o Amsterdam como un proxy web con almacenamiento en caché, la distribución seguiría al árbol. Sin embargo, esta estrategia queda corta en la práctica por tres razones. La primera razón es que los clientes en una parte dada de la red tal vez pertenezcan a distintas organizaciones, por lo que es probable que usen distintos proxies web. Recuerde que por lo general las cachés no se comparten entre organizaciones, debido al beneficio limitado de almacenar en caché cuando hay una gran cantidad de clientes, y también por cuestiones de seguridad. En segundo lugar, puede haber varias redes CDN, pero cada cliente usa sólo la caché de un solo proxy. ¿Qué CDN debe usar un cliente como su proxy? Finalmente, tal vez la cuestión más práctica de todas sea que los clientes configuran los proxies web. Pueden o no configurarse para beneficiar la distribución de contenido mediante una CDN, y hay muy poco que la CDN puede hacer al respecto.

Otra manera simple de soportar un árbol de distribución con un nivel es mediante el uso de **espejos** (*mirroring*). En este método, el servidor de origen replica el contenido sobre los nodos de la CDN como antes. A los nodos de la CDN en distintas regiones de la red se les llama **espejos**. Las páginas web en el servidor de origen contienen vínculos explícitos a los distintos espejos, que por lo general indican al usuario su ubicación. Este diseño permite al usuario seleccionar en forma manual un espejo cercano y usarlo para descargar contenido. Un uso común de los espejos es para colocar un paquete de software extenso en espejos ubicados en, por ejemplo, las costas este y oeste de Estados Unidos, Asia y Europa. Por lo general, los sitios almacenados en espejos son completamente estáticos, y la elección de sitios permanece estable durante meses o años. Son una técnica probada y demostrada. Sin embargo, dependen del usuario para realizar la distribución, ya que en realidad los espejos son sitios web distintos, incluso aunque estén vinculados entre sí.

El tercer método, que sobrepasa las dificultades de los dos métodos anteriores, usa DNS y se conoce como **redirección de DNS**. Suponga que un cliente desea obtener una página con el URL *http://www.cdn.com/pagina.html*. Para obtener esa página, el navegador usará DNS para resolver *www.cdn.com* a una dirección IP. Esta búsqueda del DNS procede de la manera usual. Mediante el uso del protocolo DNS, el navegador conoce la dirección IP del servidor de nombres para *cdn.com*, y después se contacta con el servidor de nombres para pedirle que resuelva *www.cdn.com*. Ahora viene la parte realmente inteligente. El servidor de nombres es operado por la CDN. En vez de regresar la misma dirección IP para cada solicitud, analizará la dirección IP del cliente que hace la solicitud y regresará distintas respuestas. La respuesta será la dirección IP del nodo de la CDN que está más cerca del cliente. Esto es, si un cliente en Sydney pide al servidor de nombres de la CDN que resuelva *www.cdn.com*, el servidor de nombres devolverá la dirección IP del nodo de la CDN de Sydney, pero si un cliente en Amsterdam realiza la misma solicitud, el servidor de nombres devolverá la dirección IP del nodo de la CDN de Amsterdam esta vez.

Esta estrategia es perfectamente legal, de acuerdo con la semántica del DNS. Anteriormente vimos que los servidores de nombres pueden devolver listas cambiantes de direcciones IP. Después de la resolución de nombres, el cliente de Sydney obtendrá la página de manera directa del nodo de la CDN de Sydney. Las páginas posteriores en el mismo "servidor" se obtendrán directamente del nodo de la CDN de Sydney también debido al almacenamiento en caché del DNS. En la figura 7-68 se muestra la secuencia general de los pasos.

Una pregunta compleja en el proceso anterior es qué significa encontrar el nodo de la CDN más cercano, y cómo hacerlo. Para definir "más cercano", en realidad no es la geografía lo que importa. Hay por lo menos dos factores a considerar para asociar un cliente a un nodo de la CDN. Uno de estos factores es la distancia de la red. El cliente debe tener una trayectoria de red de capacidad corta y otra alta hacia el nodo de la CDN. Esta situación producirá descargas rápidas. Las redes CDN usan un mapa que calculan previamente para realizar las traducciones entre la dirección IP de un cliente y su ubicación en la red. El

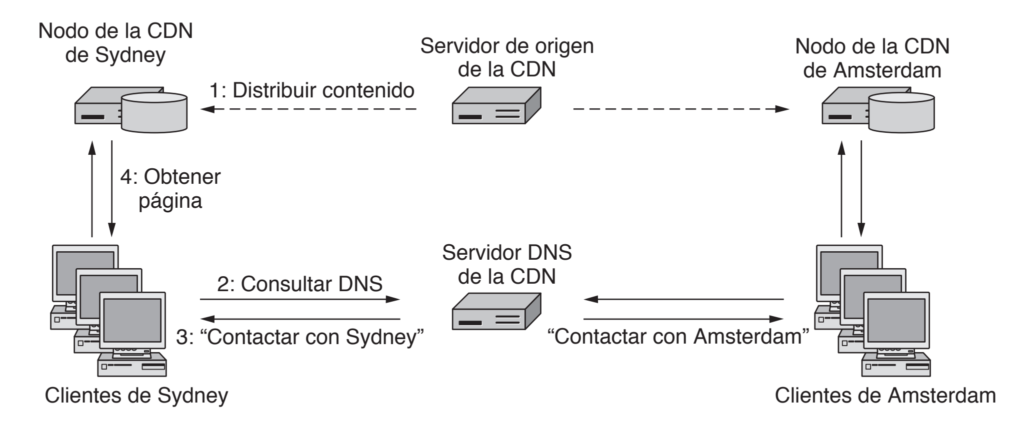


Figura 7-68. Cómo dirigir los clientes a los nodos cercanos de la CDN mediante DNS.

nodo de la CDN que se selecciona podría ser el de la distancia más corta en línea recta o tal vez no. Lo que importa es una combinación de la longitud de la trayectoria de red y cualquier límite de capacidad a lo largo de ésta. El segundo factor es la carga que ya transporta el nodo de la CDN. Si estos nodos se sobrecargan, entregarán respuestas lentas, justo igual como el servidor web sobrecargado que tratamos de evitar en primer lugar. Por ende, tal vez sea necesario balancear la carga a través de los nodos de la CDN, para lo cual se deben asociar algunos clientes a los nodos que estén un poco más alejados, pero que tengan cargas más ligeras.

Las técnicas para usar DNS en la distribución de contenido fueron practicadas por primera vez por Akamai a principios de 1998, cuando la web se empezaba a quejar de la carga de su primer crecimiento. Akamai fue la primera red CDN importante y se convirtió en líder en la industria. Tal vez algo aún más inteligente que la idea de usar DNS para conectar clientes a los nodos cercanos fue la estructura incentiva de su negocio. Las empresas pagan a Akamai para que entregue su contenido a sus clientes, de modo que tengan sitios web eficaces que a los clientes les guste usar. Los nodos de la CDN se deben colocar en ubicaciones de red con buena conectividad, lo cual en un principio significaba hacerlo dentro de las redes de los ISP. Para los ISP hay un beneficio al tener un nodo de CDN en sus redes, esto es que el nodo de CDN recorta la cantidad de ancho de banda ascendente que necesitan (y por el que deben pagar), justo igual que con las cachés de proxy. Además, el nodo de CDN mejora la capacidad de respuesta para los clientes del ISP, lo cual le hace verse bien a sus ojos y les proporciona una ventaja competitiva sobre los ISP que no tienen un nodo de CDN. Estos beneficios (a ningún costo para el ISP) hacen que instalar un nodo de CDN sea algo nada complicado para el ISP. Así, el proveedor de contenido, el ISP y los clientes se benefician, además de que la CDN gana algo de dinero. Desde 1998, otras empresas han entrado al negocio, por lo que ahora es una industria competitiva con varios proveedores.

Como lo da a entender esta descripción, la mayoría de las empresas no crean su propia CDN. En cambio, usan los servicios de un proveedor de CDN como Akamai para entregar su contenido. Para permitir que otras empresas usen el servicio de una CDN, necesitamos agregar un último paso a nuestro proceso.

Una vez que se firma el contrato para que una CDN distribuya contenido en beneficio del propietario de un sitio web, éste proporciona el contenido a la CDN. Este contenido se inserta en los nodos de la CDN. Además, el propietario vuelve a escribir cualquiera de sus páginas web que incluyan un vínculo hacia ese contenido. En vez de vincular al contenido en su sitio web, las páginas incluyen vínculos al contenido por medio de la CDN. Como ejemplo del funcionamiento de este esquema, considere el código fuente para la página web de Fluffy's Video, que se muestra en la figura 7-69(a). Después del pre-procesamiento, se transforma en la figura 7-69(b) y se coloca en el servidor de Fluffy Video como *www.fluffyvideo.com/index.html*.

```
<html>
<head> <title> Fluffy Video </title> </head>
<body>
<h1> Lista de productos de Fluffy Video </h1>
<p> Haga clic abajo para obtener muestras gratis. </p>

<a href=”koalas.mpg”> Los koalas de hoy </a> <br>
<a href=”canguros.mpg”> Canguros graciosos </a> <br>
<a href=”wombats.mpg”> Agradables wombats </a> <br>
</body>
</html>
```

(a)

```
<html>
<head> <title> Fluffy Video </title> </head>
<body>
<h1> Lista de productos de Fluffy Video </h1>
<p> Haga clic abajo para obtener muestras gratis. </p>

<a href=”http://www.cdn.com/fluffyvideo/koalas.mpg”> Los koalas de hoy </a> <br>
<a href=”http://www.cdn.com/fluffyvideo/canguros.mpg”> Canguros graciosos </a> <br>
<a href=”http://www.cdn.com/fluffyvideo/wombats.mpg”> Agradables wombats </a> <br>
</body>
</html>
```

(b)

Figura 7-69. (a) Página web original. (b) La misma página después de vincular a la CDN.

Cuando un usuario escribe el URL *www.fluffyvideo.com* en su navegador, el DNS regresa la dirección IP del propio sitio web de Fluffy Video, lo cual permite obtener la página principal (HTML) de la manera normal. Cuando el usuario hace clic en cualquiera de los hipervínculos, el navegador pide al DNS que busque *www.cdn.com*. Para esta búsqueda se contacta al servidor DNS de la CDN, quien devuelve la dirección IP del nodo CDN cercano. Después, el navegador envía una solicitud HTTP regular al nodo de la CDN; por ejemplo, para */fluffyvideo/koalas.mpg*. El URL identifica la página a devolver, e inicia la ruta con *fluffyvideo* de manera que el nodo de la CDN pueda separar las solicitudes de las distintas empresas a las que atienda. Finalmente, se devuelve el video y el usuario puede ver tiernos animalitos de peluche.

La estrategia detrás de esta división del contenido hospedado por la CDN y las páginas de entrada hospedadas por el propietario del contenido es brindar control al propietario, al tiempo que se permite a la CDN mover todo el conjunto masivo de datos. La mayoría de las páginas de entrada son pequeñas, y constan sólo de texto de HTML. Con frecuencia estas páginas incluyen vínculos a archivos extensos, como videos e imágenes. Son precisamente estos archivos extensos los que son servidos por la CDN, aun cuando el uso de una CDN sea algo totalmente transparente para los usuarios. El sitio se ve igual, pero se desempeña con mayor rapidez.

Hay otra ventaja para los sitios que usan una CDN compartida. La demanda a futuro para un sitio web puede ser difícil de predecir. Con frecuencia hay picos en la demanda, conocidos como **aglomeraciones instantáneas** (*flash crowds*). Un pico así puede ocurrir cuando se libera el producto más reciente, cuando hay un show de moda o algún otro evento, o cuando la empresa sale en las noticias. Incluso un sitio web que antes era totalmente desconocido puede convertirse de repente en el foco de Internet si es de interés para las noticias y se vincula desde los sitios populares. Como la mayoría de los sitios no están preparados para manejar incrementos masivos en el tráfico, el resultado es que muchos de ellos fallan cuando hay picos de tráfico.

Punto en cuestión. Por lo general, el sitio web de la Secretaría de Estado de Florida no es un lugar ocupado, aunque se puede buscar información sobre las corporaciones, notarías y asuntos culturales

de Florida, así como información sobre las votaciones y las elecciones en ese estado. Por cierta razón extraña, en noviembre 7 2000 (la fecha de la elección presidencial en Estados Unidos con Bush *vs*. Gore) muchas personas se interesaron de manera repentina en la página de resultados de elecciones de este sitio, el cual de repente se convirtió en uno de los sitios web más ocupados en el mundo y, naturalmente, falló como resultado. Tal vez hubiera sobrevivido si utilizara una CDN.

Mediante el uso de una CDN, un sitio tiene acceso a una capacidad de servicio de contenido muy grande. Las redes CDN más grandes tienen decenas de miles de servidores desplegados en diferentes países. Como sólo un pequeño número de sitios experimentará una aglomeración instantánea en cualquier un momento dado (por definición), esos sitios pueden usar la capacidad de la CDN de manejar la carga hasta que pase la tormenta. Es decir, la CDN puede escalar con rapidez la capacidad de servicio de un sitio.

La discusión anterior es una descripción simplificada de la forma en que funciona Akamai. Hay muchos más detalles que importan en la práctica. Los nodos de la CDN que se muestran en nuestro ejemplo son por lo general clústeres de máquinas. La redirección de DNS se realiza con dos niveles: uno para asociar clientes a la ubicación de red aproximada y otro para esparcir la carga sobre los nodos en esa ubicación. Tanto la confiabilidad como el desempeño son cuestiones importantes. Para desplazar un cliente de una máquina en un clúster a otro, las respuestas del DNS en el segundo nivel se proporcionan con valores de TTL cortos, de modo que el cliente repita la resolución después de poco tiempo. Finalmente, aunque nos hemos concentrado en la distribución de contenidos estáticos, como imágenes y videos, las redes CDN también pueden soportar la creación de páginas dinámicas, los medios de flujo continuo y otras cosas más. Para obtener más información sobre las redes CDN, consulte a Dilley y colaboradores (2002).

7.5.4 Redes de igual a igual

No todos pueden establecer una CDN con 1000 nodos en ubicaciones en todo el mundo para distribuir su contenido (en realidad no es difícil rentar 1000 máquinas virtuales alrededor del mundo, debido a la industria de hospedaje tan bien desarrollada y competitiva. Sin embargo, el proceso de establecer una CDN apenas empieza cuando se obtienen los nodos). Por suerte hay una alternativa para el resto de nosotros que es simple para usar y puede distribuir una enorme cantidad de contenido. Nos referimos a la red P2P (igual a igual).

Las redes P2P empezaron a funcionar a partir de 1999. La primera aplicación extendida fue para el crimen en masa: 50 millones de usuarios de Napster intercambiaban canciones protegidas por derechos de autor sin el permiso de los propietarios de esos derechos, hasta que la corte cerró Napster en medio de una gran controversia. Sin embargo, la tecnología de igual a igual tiene muchos usos interesantes y legales. Otros sistemas continuaron su desarrollo, con tanto interés por parte de los usuarios que el tráfico P2P pronto eclipsó al tráfico web. En la actualidad, BitTorrent es el protocolo P2P más popular. Se usa tanto para compartir videos (con licencia y del dominio público) al igual que otro tipo de contenido, que es responsable de una gran parte de todo el tráfico de Internet. Lo analizaremos en esta sección.

La idea básica de una red de compartición de archivos **P2P** (**Igual a Igual**, del inglés *Peer-to-Peer*) es que muchas computadoras se conecten entre sí y reserven sus recursos para formar un sistema de distribución de contenido. Con frecuencia las computadoras son sólo domésticas. No necesitan ser máquinas en centros de datos de Internet. A las computadoras se les llama iguales debido a que cada una de ellas puede actuar de manera alternativa como cliente para otro igual y obtener su contenido, y como servidor para proveer contenido a otros iguales. Lo que hace interesantes a los sistemas de igual a igual es que no hay una infraestructura dedicada, a diferencia de una CDN. Todos participan en la tarea de distribuir contenido, y a menudo no hay punto central de control.

Muchas personas están entusiasmadas sobre la tecnología P2P, ya que se ve como algo que otorga poder al débil. La razón no sólo es que se necesita de una empresa grande para operar una CDN, mientras que cualquiera con una computadora se puede unir a una red P2P. Resulta ser que las redes P2P tienen una capacidad formidable para distribuir contenido que puede rivalizar con el mayor de los sitios web.

Considere una red P2P compuesta de *N* usuarios promedio, cada uno de ellos con conectividad de banda ancha a 1 Mbps. La capacidad de envío agregada de la red P2P, o tasa a la que los usuarios pueden enviar tráfico a Internet, es de *N* Mbps. La capacidad de descarga, o tasa a la que los usuarios pueden recibir tráfico, es también de *N* Mbps. Cada usuario puede enviar y descargar al mismo tiempo, ya que tienen un enlace de 1 Mbps en cada dirección.

No es obvio que esto deba ser cierto, pero resulta que toda la capacidad se puede usar de manera productiva para distribuir contenido, incluso para el caso de compartir una sola copia de un archivo con todos los demás usuarios. Para ver cómo puede hacerce esto, imagine que los usuarios están organizados en un árbol binario, en donde cada usuario que no es nodo hoja envía datos a otros dos usuarios. El árbol transportará la única copia del archivo a todos los demás usuarios. Para usar el ancho de banda de envío de todos los usuarios posibles en todo momento (y por ende, distribuir el archivo grande con poca latencia), necesitamos canalizar la actividad de los usuarios en la red. Imagine que el archivo se divide en 1000 piezas. Cada usuario puede recibir una nueva pieza desde cualquier parte en un nivel superior del árbol y enviar la pieza recibida previamente hacia abajo por el árbol al mismo tiempo. De esta forma, una vez que se inicia la canalización y después de enviar un pequeño número de piezas (equivalente a la profundidad del árbol), todos los usuarios que no sean nodos hoja estarán ocupados enviando el archivo a otros usuarios. Como hay en promedio *N/2* usuarios que no son nodos hoja, el ancho de banda de envío de este árbol es de *N/2* Mbps. Podemos repetir este truco y crear otro árbol que use los otros *N/2* Mbps de ancho de banda de envío si intercambiamos los roles de los nodos hoja y los que no lo son. En conjunto, esta construcción utiliza toda la capacidad.

Este argumento significa que las redes P2P son autoescalables. Su capacidad de envío útil aumenta junto con las demandas de descarga que pueden tener sus usuarios. Siempre son lo "bastante grandes" en cierto sentido, sin la necesidad de una infraestructura dedicada. Por el contrario, hasta la capacidad de un sitio web grande es fija. Considere un sitio con sólo 100 clústeres, cada uno capaz de transmitir 10 Gbps. Esta enorme capacidad no ayuda cuando hay una cantidad pequeña de usuarios. El sitio no puede transmitir información a *N* usuarios a una tasa más rápida que *N* Mbps debido a que el límite está en los usuarios y no en el sitio web. Y cuando hay más de un millón de usuarios de 1 Mbps, el sitio web no puede transmitir datos con la suficiente rapidez como para mantener a todos los usuarios ocupados descargando. Esto puede parecer una gran cantidad de usuarios, pero las redes BitTorrent extensas (como Pirate Bay) afirman tener más de 10 000 000 usuarios. ¡Esto es más aproximado a 10 terabits/seg en términos de nuestro ejemplo!

Hay que tomar estas cifras con mucho cuidado, debido a que simplifican más de la cuenta la situación. Un reto considerable para las redes P2P es usar bien el ancho de banda cuando puede haber usuarios de todo tipo y tamaño, y pueden tener distintas capacidades de envío y descarga. Sin embargo, estas cifras indican el enorme potencial de P2P.

Hay otra razón por la que las redes P2P son importantes. Las redes CDN y otros servicios de operación central ponen a los proveedores en la posición de tener un tesoro de información personal sobre muchos usuarios, desde sus preferencias de navegación y en donde compran las personas en línea, hasta sus ubicaciones y direcciones de correo electrónico. Esta información se puede utilizar para proveer un mejor servicio y más personalizado, o se puede usar para invadir la privacidad de las personas. Esto último puede ocurrir en forma intencional (por ejemplo, como parte de un nuevo producto) o por medio de una divulgación accidental que comprometa esa información. En los sistemas P2P no puede haber un solo proveedor que sea capaz de monitorear todo el sistema completo. Esto no necesariamente significa que los sistemas P2P provean privacidad, ya que los usuarios confían entre sí hasta cierto grado. Sólo

significa que pueden proveer una manera distinta de privacidad a la de los sistemas administrados en forma central. En la actualidad se están explorando los sistemas P2P para ofrecer servicios más allá de la compartición de archivos (por ejemplo, almacenamiento, flujo continuo); el tiempo dirá si esta ventaja es importante o no.

La tecnología P2P ha seguido dos trayectorias relacionadas en el transcurso de su desarrollo. En el lado más práctico están los sistemas que se utilizan a diario. Los más conocidos de estos sistemas se basan en el protocolo BitTorrent. En el lado más académico, ha existido un intenso interés en los algoritmos DHT (Tabla de Hash Distribuida) que permiten a los sistemas P2P funcionar bien como un todo, pero que para nada dependen de componentes centralizados. A continuación analizaremos ambas tecnologías.

BitTorrent

Braham Cohen desarrolló el protocolo BitTorrent en 2001 para permitir que un conjunto de iguales compartieran archivos con rapidez y facilidad. Existen docenas de clientes disponibles sin costo, los cuales se comunican mediante este protocolo, así como hay muchos navegadores que se comunican con los servidores web mediante el protocolo HTTP. El protocolo está disponible como un estándar abierto en *www.bittorrent.org*.

En un sistema típico de igual a igual, como el que se forma con BitTorrent, cada uno de los usuarios tiene cierta información que puede ser de interés para los otros usuarios. Esta información puede ser software libre, música, videos, fotografías, etc. Hay tres problemas que necesitan ser resueltos para compartir contenido en esta configuración:

1. ¿Cómo encuentra un igual a otros iguales que tienen el contenido que desea descargar?
2. ¿Cómo replican los iguales el contenido para ofrecer descargas de alta velocidad para todos?
3. ¿Cómo se animan los iguales unos a otros para enviar contenido a alguien más, así como descargar contenido para ellos mismos?

El primer problema existe debido a que no todos los iguales tendrán todo el contenido, por lo menos al principio. La metodología utilizada en BitTorrent es que cada proveedor de contenido debe crear una descripción de contenido, conocida como **torrent**. El torrent es mucho más pequeño que el contenido, y es usado por un igual para verificar la integridad de los datos que descarga de otros iguales. Otros usuarios que deseen descargar el contenido deben obtener primero el torrent; por ejemplo, pueden encontrarlo en una página web que anuncie el contenido.

El torrent es tan sólo un archivo en un formato especificado que contiene dos tipos de información. Uno de ellos es el nombre de un **rastreador** (*tracker*): un servidor que conduce a los iguales al contenido del torrent. El otro tipo de información es una lista de piezas de igual tamaño, o **trozos** (*chunks*), que forman el contenido. Se pueden usar distintos tamaños de trozos para los diferentes torrents, por lo general de 64 KB a 512 KB. El archivo torrent contiene el nombre de cada trozo, el cual se proporciona como un *hash* SHA-1 de 160 bits del trozo. En el capítulo 8 hablaremos sobre hashes criptográficos, como SHA-1. Por ahora puede considerar un hash como una suma de verificación más larga y segura. Dado el tamaño de los trozos y los hashes, el archivo torrent es por lo menos 30 veces más pequeño que el contenido, por lo que se puede transferir con rapidez.

Para descargar el contenido descrito en un torrent, un igual primero se contacta con el rastreador de ese torrent. El **rastreador** es un servidor que mantiene una lista de todos los demás iguales que están descargando y enviando el contenido en forma activa. A este conjunto de iguales se le denomina **enjambre** (*swarm*). Los miembros del enjambre se contactan con el rastreador de manera regular para informarle que siguen activos, así como hacerle saber cuándo se salen del enjambre. Cuando un nuevo igual se con-

tacta con el rastreador para unirse al enjambre, el rastreador le indica sobre los demás iguales que hay en el enjambre. Obtener el torrent y contactarse con el rastreador son los primeros dos pasos para descargar contenido, como se muestra en la figura 7-70.

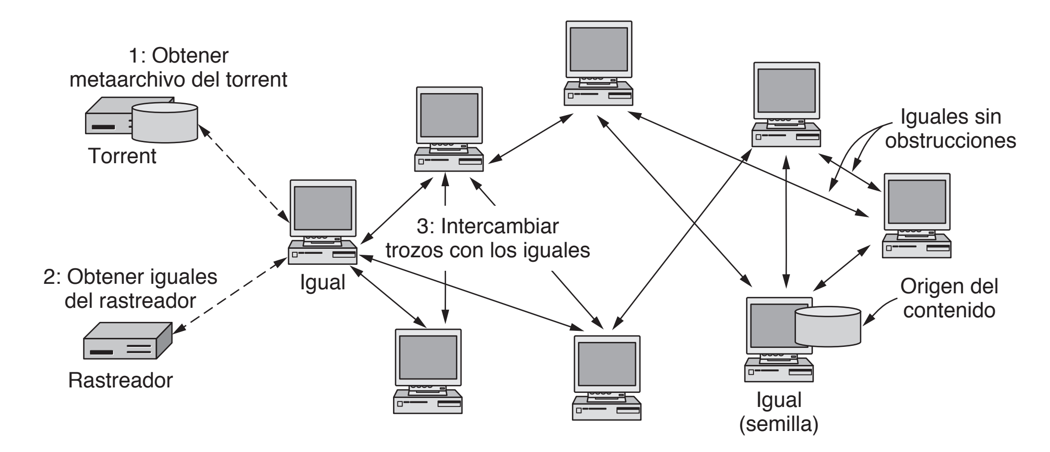


Figura 7-70. BitTorrent.

El segundo problema es cómo compartir el contenido de una manera que proporcione descargas rápidas. Cuando se forma un enjambre por primera vez, algunos iguales deben tener todos los trozos que forman el contenido. A estos iguales se les llama **sembradores** (*seeders*). Otros iguales que se unan al enjambre no tendrán trozos; son los iguales que van a descargar el contenido.

Mientras un igual participa en un enjambre, al mismo tiempo descarga los trozos que le faltan de los otros iguales, y envía los trozos que tiene a otros iguales que los necesitan. Este intercambio se muestra como el último paso de distribución de contenido en la figura 7-70. Con el tiempo, el igual recopila más trozos hasta que haya descargado todo el contenido. El igual puede salir del enjambre (y regresar) en cualquier momento. Por lo general un igual permanecerá durante un corto tiempo después de terminar su propia descarga. Como entran y salen iguales, la tasa de renovación en un enjambre puede ser bastante alta.

Para que el método anterior funcione bien, cada trozo debe estar disponible en muchos iguales. Si todos obtuvieran los trozos en el mismo orden, es probable que muchos iguales dependieran de los sembradores para el siguiente trozo. Esto crearía un cuello de botella. En cambio, los iguales intercambian listas de los trozos que tienen unos con otros. Después seleccionan trozos raros, que son difíciles de encontrar para descargarlos. La idea es que al descargar un trozo raro se haga una copia del mismo, lo cual a su vez hará que el trozo sea más fácil de encontrar y descargar para los otros iguales. Si todos los iguales hacen esto, después de un tiempo corto todos los trozos estarán disponibles ampliamente.

Tal vez el tercer problema es el más interesante. Los nodos de la CDN se establecen de manera exclusiva para ofrecer contenido a los usuarios, pero los nodos P2P no. Son computadoras de los usuarios, y éstos pueden estar más interesados en obtener una película que en ayudar a otros usuarios con sus descargas. Los nodos que toman recursos de un sistema sin contribuir en especie se denominan **viajeros sin boleto** (*free-riders*) o **sanguijuelas** (*leechers*). Si hay demasiados de éstos, el sistema no funcionará bien. Se sabía que los primeros sistemas P2P los hospedaban (Saroiu y colaboradores, 2003), por lo que BitTorrent buscó minimizarlos.

La metodología que se lleva a cabo en los clientes de BitTorrent es recompensar a los iguales que muestran un buen comportamiento de envío. Cada igual muestrea en forma aleatoria a los otros iguales, y obtiene

trozos de ellos mientras les envía otros trozos. El igual continúa intercambiando trozos con sólo un pequeño número de iguales que proveen el desempeño de descarga más alto, mientras que también prueban al azar con otros iguales para que encuentren buenos socios. Probar iguales al azar es un proceso que permite a los recién llegados obtener trozos iniciales que pueden intercambiar con otros iguales. Se dice que los iguales con los que un nodo se encuentra intercambiando trozos son **sin obstrucciones** (*unchoked*).

Con el tiempo, este algoritmo tiene la intención de asociar a los iguales con tasas de envío y descarga comparables entre sí. Entre más contribuya un igual con los otros iguales, más podrá esperar a cambio. Al usar un conjunto de iguales también es conveniente saturar el ancho de banda de descarga de un igual para obtener un alto desempeño. En cambio, si un igual no envía trozos a otros iguales, o si lo hace con mucha lentitud, será desconectado u **obstruido** (*choked*) tarde o temprano. Esta estrategia está en contra del comportamiento antisocial en el que los iguales viajan gratis en el enjambre.

Algunas veces el algoritmo de obstrucción se describe como una implementación de la estrategia de **"ojo por ojo"** (*tit-for-tat*) que fomenta la cooperación en interacciones repetidas. Sin embargo, no evita que los clientes jueguen con el sistema en ningún sentido fuerte (Piatek y colaboradores, 2007). No obstante, la atención a los problemas y los mecanismos que dificultan más el hecho de que los usuarios casuales viajen de manera gratuita han contribuido de igual forma al éxito de BitTorrent.

Como puede ver de nuestro análisis, BitTorrent incluye un extenso vocabulario. Hay torrents, enjambres, sanguijuelas, sembradores y rastreadores, así como rechazos, obstrucciones, acechos, y más. Para obtener más información consulte el artículo corto sobre BitTorrent (Cohen, 2003) y busque en la web, empezando con *www.bittorrent.org*.

DHT: Tablas de Hash Distribuidas

El surgimiento de las redes de compartición de archivos P2P alrededor del año 2000 generó mucho interés en la comunidad de investigación. La esencia de los sistemas P2P es que evitan las estructuras administradas en forma central de las redes CDN y otros sistemas. Ésta puede ser una ventaja considerable. Los componentes administrados en forma central se convierten en un "cuello de botella", a medida que el sistema crece a un tamaño muy grande, además de que constituyen un solo punto de falla. Los componentes centrales también se pueden usar como punto de control (por ejemplo, para apagar la red P2P). Sin embargo, los primeros sistemas P2P sólo estaban descentralizados en parte o, si estaban totalmente descentralizados, eran ineficientes.

La forma tradicional de BitTorrent que acabamos de describir usa transferencias de igual a igual y un rastreador centralizado para cada enjambre. Es el rastreador el que resulta ser la parte más difícil de descentralizar en un sistema de igual a igual. El problema clave es cómo averiguar qué iguales tienen contenido específico que se esté buscando. Por ejemplo, cada usuario podría tener uno o más elementos de datos como canciones, fotografías, programas, archivos y otras cosas que otros usuarios tal vez quieran leer. ¿Cómo encuentran los otros usuarios estos elementos? Es sencillo crear un índice de lo que cada quien posee, pero es un método centralizado. Tampoco ayuda el hecho de hacer que cada igual mantenga su propio índice. Ciertamente es distribuido. Sin embargo, se requiere tanto trabajo para mantener los índices de todos los iguales actualizados (puesto que el contenido se mueve alrededor del sistema), que no vale la pena el esfuerzo.

La pregunta abordada por la comunidad de investigación trataba acerca de si era posible crear índices P2P que fueran totalmente distribuidos pero que se desempeñaran bien. En cuanto al buen desempeño, nos referimos a tres cosas. Primero, cada nodo mantiene sólo una pequeña cantidad de información sobre los otros nodos. Esta propiedad significa que no será costoso mantener el índice actualizado. Segundo, cada nodo puede buscar entradas en el índice con rapidez. De cualquier otro modo, no sería un índice muy útil. Tercero, cada nodo puede usar el índice al mismo tiempo, incluso a medida que entran y salen otros nodos. Esta propiedad significa que el desempeño del índice aumenta con el número de nodos.

La respuesta a la pregunta fue: "Sí". Se inventaron cuatro soluciones distintas en 2001. Éstas son: Chord (Stoica y colaboradores, 2001), CAN (Ratnasamy y colaboradores, 2001), Pastry (Rowstron y Druschel, 2001) y Tapestry (Zhao y colaboradores, 2004). Otras soluciones se inventaron poco tiempo después, incluyendo Kademlia, que se utiliza en la práctica (Maymounkov y Mazieres, 2002). Estas soluciones se conocen como tablas **DHT** (**Tablas de Hash Distribuidas**, del inglés *Distributed Hash Tables*) debido a que la funcionalidad básica de un índice es asociar una clave con un valor. Esto es como una tabla de hash y, desde luego, las soluciones son versiones distribuidas.

Para hacer su trabajo, las tablas DHT imponen una estructura regular sobre la comunicación entre los nodos, como veremos más adelante. Este comportamiento es muy distinto al de las redes P2P tradicionales que usan las conexiones que realicen los iguales. Por esta razón, a las tablas DHT se les conoce como **redes P2P estructuradas**. Los protocolos P2P tradicionales construyen **redes P2P no estructuradas**.

La solución de DHT que describiremos es Chord. Como escenario, considere cómo reemplazar el rastreador centralizado que se utiliza de manera tradicional en BitTorrent, por un rastreador totalmente distribuido. Se puede usar Chord para resolver este problema. En este escenario, el índice general es un listado de todos los enjambres a los que se puede unir una computadora para descargar contenido. La clave que se utiliza para buscar en el índice es la descripción del torrent del contenido. Esta descripción identifica en forma única a un enjambre del que se puede descargar contenido como los hashes de todos los trozos de contenido. El valor almacenado en el índice para cada clave es la lista de iguales que conforman el enjambre. Estos iguales son las computadoras con las que se debe uno contactar para descargar el contenido. Una persona que desee descargar una película, sólo tiene la descripción del torrent. La pregunta que debe responder la DHT es: ¿cómo, sin tener una base de datos central, puede una persona averiguar qué iguales (de los millones de nodos de BitTorrent) va a usar para descargar la película?

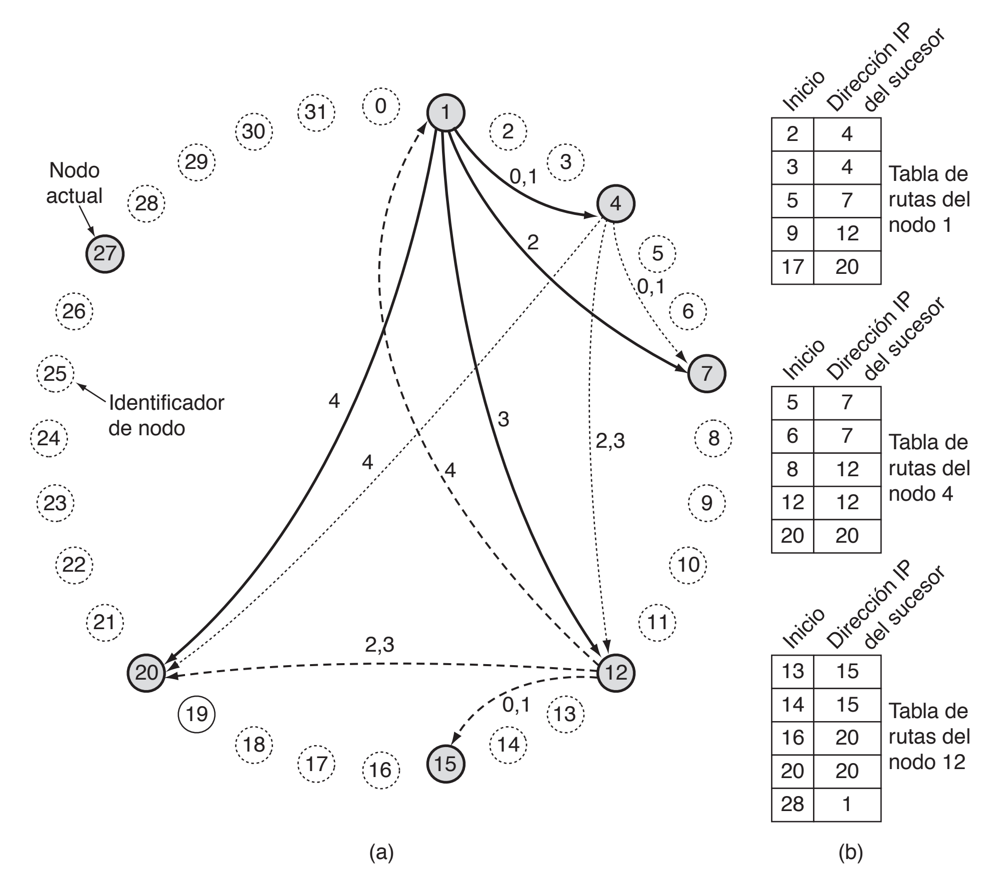
Una DHT de Chord consiste en n nodos participantes. Éstos son nodos que ejecutan BitTorrent en nuestro escenario. Cada nodo tiene una dirección IP mediante la cual se puede contactar. El índice general se esparce a través de los nodos. Esto implica que cada nodo almacena bits y piezas del índice para que lo usen otros nodos. La parte clave de Chord es que navega por el índice mediante el uso de identificadores en un espacio virtual, no por las direcciones IP de los nodos ni los nombres de contenido, como en las películas. En concepto, los identificadores son simplemente números de m bits que se pueden ordenar en forma ascendente en un anillo.

Para convertir la dirección de un nodo en un identificador, se asocia con un número de m bits mediante el uso de una función de hash llamada *hash*. Chord usa SHA-1 para *hash*. Es el mismo hash que mencionamos al describir BitTorrent. Lo analizaremos cuando hablemos sobre la criptografía en el capítulo 8. Por ahora basta con decir que es sólo una función que recibe una cadena de bytes de longitud variable como argumento, y produce un número de 160 bits altamente aleatorio. Por lo tanto, podemos usarla para convertir cualquier dirección IP en un número de 160 bits, al cual se le conoce como **identificador de nodo**.

En la figura 7-71(a), mostramos el círculo del identificador de nodo para $m = 5$ (sólo ignore los arcos de la parte de en medio por el momento). Algunos de los identificadores corresponden a los nodos, pero la mayoría no. En este ejemplo, los nodos con los identificadores 1, 4, 7, 12, 15, 20 y 27 corresponden a nodos actuales y están sombreados en la figura; el resto no existe.

Ahora definiremos la función *sucesor*(k) como el identificador del primer nodo actual que va después de k alrededor del círculo, en sentido de las manecillas del reloj. Por ejemplo, *sucesor* (6) = 7, *sucesor* (8) = 12, y *sucesor* (22) = 27.

Una **clave** también se produce al convertir un nombre de contenido con la función *hash* (es decir, SHA-1) para generar un número de 160 bits. En nuestro escenario, el nombre del contenido es el torrent. Así, para convertir *torrent* (el archivo de descripción del torrent) en su clave, calculamos *clave* = *hash* (*torrent*). Este cálculo es sólo una llamada a un procedimiento local para *hash*.



Inicio	Dirección IP del sucesor
2	4
3	4
5	7
9	12
17	20

Inicio	Dirección IP del sucesor
5	7
6	7
8	12
12	12
20	20

Inicio	Dirección IP del sucesor
13	15
14	15
16	20
20	20
28	1

Figura 7-71. (a) Un conjunto de 32 identificadores de nodo distribuidos en un círculo. Los sombreados corresponden a las máquinas reales. Los arcos muestran las rutas de los nodos 1, 4 y 12. Las etiquetas en los arcos son los índices de tabla. (b) Ejemplos de las tablas de rutas.

Para empezar un nuevo enjambre, un nodo necesita insertar un nuevo par clave-valor que consista de (*torrent*, *mi-dirección-IP*) en el índice. Para lograr esto, el nodo pide a *sucesor*(*hash*(*torrent*)) que almacene *mi-dirección-IP*. De esta forma, el índice se distribuye sobre los nodos al azar. Para la tolerancia a fallas, se pueden usar *p* funciones de hash distintas para almacenar los datos a *p* nodos, pero no profundizaremos aquí sobre el tema de la tolerancia a fallas.

Tiempo después de que se construye la DHT, otro nodo desea buscar un torrent para unirse al enjambre y descargar contenido. Para buscar el *torrent*, el nodo primero le aplica la función hash para obtener la *clave*, y después usa *sucesor (clave)* para encontrar la dirección IP del nodo que almacena el valor correspondiente. El valor es la lista de iguales en el enjambre; el nodo puede agregar su dirección IP a la lista y contactarse con los otros iguales para descargar contenido con el protocolo BitTorrent.

El primer paso es sencillo; el segundo no lo es tanto. Para que sea posible encontrar la dirección IP del nodo correspondiente a cierta clave, cada nodo tiene que mantener ciertas estructuras de datos administrativas. Una de éstas es la dirección IP de su nodo sucesor a lo largo del círculo de identificadores de nodo. Por ejemplo, en la figura 7-71, el sucesor del nodo 4 es 7 y el sucesor del nodo 7 es 12.

Ahora la búsqueda puede proceder de la siguiente manera. El nodo solicitante envía un paquete a su sucesor que contiene su dirección IP y la clave que está buscando. El paquete se propaga a lo largo del anillo hasta que localiza el sucesor para el identificador de nodo que está buscando. Ese nodo verifica si

tiene alguna información que coincida con la clave y, de ser así, lo devuelve directamente al nodo solicitante, cuya dirección IP tiene de antemano.

Sin embargo, una búsqueda lineal de todos los nodos es muy ineficiente en un sistema de igual a igual grande, ya que el número promedio de nodos requeridos por búsqueda es de $n/2$. Para agilizar de manera considerable la búsqueda, cada nodo también mantiene lo que Chord llama una **tabla de rutas**. La tabla de rutas tiene m entradas, indexadas desde el 0 hasta $m - 1$, cada una de las cuales apunta a un nodo actual distinto. Cada entrada tiene dos campos: *inicio* y la dirección IP de *sucesor (inicio)*, como se muestra en los tres nodos de ejemplo de la figura 7-71(b). Los valores de los campos para la entrada i en un nodo con el identificador k son:

$Inicio = k + 2^i$ (*módulo* 2^m)
Dirección IP de *sucesor*(*inicio*[i])

Cabe mencionar que cada nodo almacena las direcciones IP de un número relativamente pequeño de nodos, y que la mayoría de éstos son bastante cercanos en términos del identificador de nodo.

Si usamos la tabla de rutas, la búsqueda de *clave* en el nodo k se realiza de la siguiente manera. Si *clave* está entre k y *sucesor* (k), el nodo que contiene información sobre *clave* es *sucesor* (k) y la búsqueda termina. En caso contrario, se busca en la tabla de rutas para encontrar la entrada cuyo campo *inicio* sea el predecesor más cercano de *clave*. Después se envía una solicitud directamente a la dirección IP en esa entrada de la tabla de rutas para pedirle que continúe con la búsqueda. Como está más cerca de *clave* pero sigue debajo de ella, es muy probable que pueda regresar la respuesta con sólo un pequeño número de consultas adicionales. De hecho, como cada búsqueda divide a la mitad la distancia restante al destino, se puede mostrar que el número promedio de búsquedas es $\log_2 n$.

Como primer ejemplo, considere buscar la *clave* = 3 en el nodo 1. Como el nodo 1 sabe que el 3 está entre él y su sucesor, 4, el nodo deseado es 4 y la búsqueda termina; se devuelve la dirección IP del nodo 4.

Como segundo ejemplo, considere buscar la *clave* = 16 en el nodo 1. Como 16 no está entre 1 y 4, se consulta la tabla de rutas. El predecesor más cercano a 16 es 9, por lo que la solicitud se reenvía a la dirección IP de la entrada de 9, es decir, la del nodo 12. El nodo 12 tampoco sabe la respuesta por sí solo, así que busca el nodo predecesor más cercano a 16 y encuentra el 14, que produce la dirección IP del nodo 15. A continuación se envía una consulta ahí. El nodo 15 observa que el 16 está entre él y su sucesor (20), por lo que devuelve la dirección IP de 20 al que hizo la llamada, que regresa al nodo 1.

Como los nodos se unen y se separan todo el tiempo, Chord necesita una manera de manejar estas operaciones. Suponemos que cuando el sistema empezó a operar era lo bastante pequeño como para que los nodos sólo tuvieran que intercambiar información de manera directa para construir el primer círculo y las tablas de rutas. Después de eso, se requiere un procedimiento automatizado. Cuando un nuevo nodo r se desea unir, debe contactarse con un nodo existente y pedirle que busque la dirección IP del *sucesor* (r) para él. A continuación el nuevo nodo pregunta a *sucesor* (r) por su predecesor. Después el nuevo nodo pide a estos dos últimos que inserten r entre ellos en el círculo. Por ejemplo, si el nodo 24 de la figura 7-71 desea unirse, pide a cualquier nodo que busque el *sucesor* (24), que es 27. Después pide a 27 su predecesor (20). Luego indica a estos dos últimos sobre su existencia, 20 usa a 24 como su sucesor y 27 usa a 24 como su predecesor. Además, el nodo 27 entrega las claves en el rango 21-24, que ahora pertenecen a 24. En este punto, 24 se inserta por completo.

Sin embargo, ahora muchas tablas de rutas están mal. Para corregirlas, cada nodo ejecuta un proceso en segundo plano que recalcula de manera periódica cada ruta mediante una llamada a *sucesor*. Cuando una de estas consultas llega a un nuevo nodo, se actualiza la entrada de la ruta correspondiente.

Cuando un nodo sale sin problemas, entrega sus claves a su sucesor e informa a su predecesor sobre su partida, de modo que éste pueda crear un enlace al sucesor del nodo que va a salir. Cuando un nodo

falla, surge un problema debido a que su predecesor ya no tiene un sucesor válido. Para mitigar este problema, cada nodo lleva el registro no sólo de su sucesor directo, sino también de sus s sucesores directos, para que pueda ignorar hasta $s - 1$ nodos que fallen en forma consecutiva y pueda volver a reconectar el círculo en caso de desastre.

Se han realizado innumerables investigaciones sobre las DHT desde que aparecierón. Para dar al lector una idea de cuántas, le haremos una pregunta: ¿cuál es el artículo sobre redes más citado de todos los tiempos? Tendrá dificultades para encontrar un artículo que se haya citado más que el artículo fundamental de Chord (Stoica y colaboradores, 2001). A pesar de esta auténtica montaña de investigación, las aplicaciones de las DHT apenas si comienzan a emerger con lentitud. Algunos clientes de BitTorrent utilizan tablas DHT para proveer un rastreador totalmente distribuido del tipo que describimos antes. Los grandes servicios de nubes comerciales, como Dynamo de Amazon, también incorporan técnicas de DHT (DeCandia y colaboradores, 2007).

7.6 RESUMEN

El uso de nombres en ARPANET empezo de una manera muy simple: un archivo de texto ASCII listaba los nombres de todos los hosts y sus correspondientes direcciones IP. Cada noche, todas las máquinas descargaban este archivo. Pero cuando ARPANET mutó a Internet y se expandió en forma explosiva, se requería un esquema de nombres mucho más sofisticado y dinámico. El que se utiliza ahora es un esquema jerárquico llamado Sistema de nombres de dominio. Organiza todas las máquinas en Internet en un conjunto de árboles. En el nivel superior están los dominios genéricos conocidos, incluidos *com* y *edu*, así como cerca de 200 dominios de países. El DNS se implementa como una base de datos distribuida con servidores en todo el mundo. Al consultar un servidor DNS, un proceso puede asociar un nombre de dominio de Internet a la dirección IP que se utiliza para comunicarse con una computadora para ese dominio.

El correo electrónico es la aplicación revolucionaria original de Internet. Todo el mundo lo sigue utilizando, desde niños pequeños hasta abuelos. La mayoría de los sistemas de correo electrónico en el mundo usan el sistema de correo que se define ahora en los RFC 5321 y 5322. Los mensajes tienen encabezados ASCII simples y se pueden enviar muchos tipos de contenido mediante el uso de MIME. El correo se envía a los agentes de transferencia de mensajes para que los entreguen y se obtienen de estos agentes para que los presente una variedad de agentes de usuario, incluyendo las aplicaciones web. El correo enviado se entrega mediante el protocolo SMTP, el cual trabaja realizando una conexión TCP desde el agente de transferencia de mensajes emisor hasta el receptor.

La web es la aplicación que la mayoría de las personas consideran como Internet. En un principio era un sistema para enlazar páginas de hipertexto (escritas en HTML) de manera transparente entre varias máquinas. Para descargar las páginas se crea una conexión TCP del navegador a un servidor y se usa HTTP. Hoy en día, gran parte del contenido en web se produce en forma dinámica, ya sea en el servidor (por ejemplo, mediante PHP) o en el navegador (por ejemplo, con JavaScript). Cuando se combinan con bases de datos de *back-end*, las páginas dinámicas de servidor permiten aplicaciones web como comercio electrónico y búsqueda. Las páginas dinámicas de navegador están evolucionando en aplicaciones completas, como el correo electrónico, que se ejecutan dentro del navegador y usan los protocolos web para comunicarse con los servidores remotos.

El almacenamiento en caché y las conexiones persistentes se utilizan mucho para mejorar el desempeño web. Puede ser todo un reto usar la web en dispositivos móviles, a pesar del aumento del ancho de banda y el poder de procesamiento de estos dispositivos. Con frecuencia los sitios web envían versiones especiales de páginas con imágenes más pequeñas y una navegación menos compleja para dispositivos con pantallas pequeñas.

Los protocolos web se utilizan cada vez más para la comunicación de una máquina a otra. Se prefiere el XML en vez del HTML para describir el contenido de una manera fácil de procesar para las máquinas. SOAP es un mecanismo de RPC que envía mensajes XML mediante el uso de HTTP.

El audio y el video digital han sido impulsores clave para Internet desde el año 2000. La mayor parte del tráfico de Internet en la actualidad es video. Gran parte de éste se transmite mediante flujo continuo desde sitios web a través de una mezcla de protocolos (incluyendo RTP/UDP y RTP/HTTP/TCP). Los medios en vivo se transmiten en flujo continuo a muchos consumidores. Se incluyen la radio de Internet y las estaciones de TV que difunden todo tipo de eventos. El audio y el video también se utilizan para las conferencias en tiempo real. Muchas llamadas usan voz sobre IP en vez de la red telefónica tradicional, incluyendo las videoconferencias.

Hay una pequeña cantidad de sitios web con una enorme popularidad, así como una gran cantidad de sitios menos populares. Para dar servicio a los sitios populares se han implementado redes de distribución de contenido. Las redes CDN usan DNS para dirigir a los clientes a un servidor cercano; los servidores se colocan en centros de datos en todo el mundo. Como alternativa, las redes P2P permiten que una colección de máquinas comparta contenido, como películas, entre ellas mismas. Estas redes proveen una capacidad de distribución de contenido que escala con el número de máquinas en la red P2P y que son capaces de competir con el más grande de los sitios.

PROBLEMAS

1. Muchas computadoras de negocios tienen tres identificadores distintos y únicos en todo el mundo. ¿Cuáles son?

2. En la figura 7-4, no hay punto después de *laserjet*. ¿Por qué no?

3. Considere una situación en la que un ciberterrorista hace que todos los servidores DNS en el mundo fallen al mismo tiempo. ¿Cómo cambiaría esto la habilidad de un usuario de usar Internet?

4. DNS usa UDP en vez de TCP. Si se pierde un paquete DNS, no hay recuperación automática. ¿Provoca esto un problema y, de ser así, cómo se resuelve?

5. John quiere tener un nombre de dominio original y usa un programa de aleatorización para generar un segundo nombre de domino. Desea registrar este nombre de dominio en el domino genérico *com*. El nombre de dominio que se generó tiene una longitud de 253 caracteres. ¿Permitirá el registrador de *com* registrar este dominio?

6. ¿Puede una máquina con un solo nombre DNS tener varias direcciones IP? ¿Cómo podría ocurrir esto?

7. El número de empresas con un sitio web ha crecido de manera explosiva en años recientes. Como resultado, miles de empresas están registradas en el domino *com*, lo cual provoca una carga pesada en el servidor de nivel superior de este dominio. Sugiera una forma de mitigar este problema sin cambiar el esquema de nombres (es decir, sin introducir nuevos nombres de dominio de nivel superior). Es permitido que su solución requiera cambios en el código cliente.

8. Algunos sistemas de correo electrónico soportan un campo de encabezado *Content Return*:. Este campo especifica si se va a regresar o no el cuerpo de un mensaje en caso de que no se entregue. ¿Pertenece este campo a la envoltura o al encabezado?

9. Los sistemas de correo electrónico necesitan directorios para poder buscar las direcciones de correo electrónico de las personas. Para crear dichos directorios, hay que descomponer los nombres en componentes estándar (por ejemplo, primer nombre, apellido paterno) para que sea posible realizar búsquedas. Mencione algunos problemas que se deben resolver para que un estándar mundial sea aceptable.

10. Un gran despacho jurídico, que cuenta con muchos empleados, proporciona una sola dirección de correo electrónico a cada empleado, de la siguiente manera: *<inicio_sesión>@despachojuridico.com*. Sin embargo, el despacho no define de manera explícita el formato del inicio de sesión. Por ende, algunos empleados usan su

nombre de pila como inicio de sesión, algunos su apellido, otros sus iniciales, etc. La empresa desea ahora crear un formato fijo; por ejemplo:

nombrepila.apellido@despachojuridico.com,

que se pueda usar para las direcciones de todos sus empleados. ¿Cómo se puede hacer esto sin que haya tanto alboroto?

11. Un archivo binario tiene 4 560 bytes de longitud. ¿Qué longitud tendrá si se codifica mediante la codificación base64, con un par CR+LF insertado después de cada 110 bytes enviados y al final?

12. Nombre cinco tipos MIME que no se hayan mencionado en este libro. Puede verificar con su navegador en Internet para obtener más información.

13. Suponga que desea enviar un archivo MP3 a un amigo, pero el ISP de su amigo limita el tamaño de cada mensaje entrante a 1 MB y el archivo MP3 es de 4 MB. ¿Hay alguna forma de manejar esta situación mediante el uso del RFC 5322 y de MIME?

14. Suponga que John acaba de establecer un mecanismo de reenvío automático en la dirección de correo electrónico de su trabajo, la cual recibe todos los correos relacionados con la empresa, para reenviarlos a su dirección de correo personal, que comparte con su esposa. La esposa de John no estaba al tanto de esto, por lo que activó un agente de vacaciones en su cuenta personal. Como John reenvió su correo electrónico, no estableció un daemon de vacaciones en su máquina de trabajo. ¿Qué ocurre cuando se recibe un correo electrónico en la dirección del trabajo de John?

15. En cualquier estándar, como el RFC 5 322, se requiere una gramática precisa de lo permitido, de modo que las distintas implementaciones puedan interfuncionar. Incluso los elementos simples se tienen que definir con cuidado. Los encabezados SMTP permiten espacio en blanco entre los tokens. Mencione *dos* definiciones alternativas plausibles de espacio blanco entre tokens.

16. ¿Es el agente de vacaciones parte del agente de usuario, o parte del agente de transferencia de mensajes? Desde luego que se establece mediante el agente de usuario pero, ¿es el agente de usuario el que realmente envía las respuestas? Explique su respuesta.

17. En una versión simple del algoritmo de Chord para la búsqueda de igual a igual, las búsquedas no usan la tabla de rutas. En vez de ello, son lineales alrededor del círculo, en cualquier dirección. ¿Puede un nodo predecir con precisión en qué dirección debe buscar? Explique su respuesta.

18. IMAP permite a los usuarios obtener y descargar el correo electrónico de un buzón remoto. ¿Significa esto que el formato interno de los buzones de correo se tiene que estandarizar, de modo que cualquier programa IMAP del lado cliente pueda leer el buzón de correo en cualquier servidor de correo? Explique su respuesta.

19. Considere el círculo de Chord de la figura 7-71. Suponga que el nodo 18 de repente se conecta a Internet. ¿Cuál de las tablas de rutas que se muestran en la figura se ve afectada? ¿Cómo?

20. ¿Utiliza el correo web POP3, IMAP o ninguna de las dos tecnologías? Si usa una de ellas, ¿por qué se eligió esa? Si no utiliza ninguna, ¿a cuál de las dos se parece más?

21. Cuando se envían páginas web, se anteponen encabezados MIME. ¿Por qué?

22. ¿Es posible que cuando un usuario haga clic en un vínculo dentro de Firefox se inicie un ayudante específico, pero que al hacer clic en el mismo vínculo en Internet Explorer se inicie un ayudante completamente distinto, aun cuando el tipo MIME devuelto en ambos casos sea idéntico? Explique su respuesta.

23. Aunque no se mencionó en el texto, una forma alternativa para un URL es usar la dirección IP en vez de su nombre DNS. Use esta información para explicar por qué un nombre DNS no puede terminar con un dígito.

24. Imagine que alguien en el Departamento de Matemáticas, en Stanford, acaba de escribir un nuevo documento, incluyendo una prueba que desea distribuir por FTP para que sus colegas la revisen. Coloca el programa en el directorio *ftp/pub/paraRevision/nuevaPrueba.pdf de FTP*. ¿Cuál sería el probable URL para este programa?

25. En la figura 7-22, *www.aportal.com* lleva el registro de las preferencias de usuario en una cookie. Una desventaja de este esquema es que las cookies están limitadas a 4 KB, por lo que si las preferencias son extensas, por ejemplo: muchas acciones de valores, equipos deportivos, tipos de noticias, información meteorológica

para varias ciudades, especiales en muchas categorías de productos, y más, tal vez se llegue al límite de 4 KB. Diseñe una manera alternativa para mantener el registro de las preferencias que no tenga este problema.

26. El banco Sloth desea facilitar las operaciones bancarias en línea para sus clientes con poco iniciativa, de modo que después de que un cliente inicie sesión y se autentifique mediante una contraseña, el banco devuelva una cookie que contenga un número de ID de cliente. De esta forma, el cliente no tendrá que identificarse a sí mismo o escribir una contraseña en sus visitas posteriores al banco en línea. ¿Qué piensa de esta idea? ¿Funcionará? ¿Es una buena idea?

27. (a) Considere la siguiente etiqueta de HTML:

```
<h1 title="este es el encabezado"> ENCABEZADO 1 </h1>
```

¿Bajo qué condiciones usa el navegador el atributo *TITLE*, y cómo?
(b) ¿En qué difieren el atributo *TITLE* y el atributo *ALT*?

28. ¿Cómo puede crear una imagen en la que se pueda hacer clic mediante HTML? Dé un ejemplo.

29. Escriba una página de HTML que incluya un vínculo a la dirección de correo electrónico *nombreusuario@NombreDominio.com*. ¿Qué ocurre cuando un usuario hace clic en este vínculo?

30. Escriba una página XML para el registrador de una universidad que liste a varios estudiantes, cada uno de los cuales con su nombre, dirección y GPA.

31. Para cada una de las siguientes aplicaciones, indique si sería (1) posible y (2) mejor usar una secuencia de comandos de PHP o de JavaScript, y por qué:

(a) mostrar un calendario para cualquier mes solicitado a partir de septiembre de 1752.
(b) mostrar el programa de vuelos de Amsterdam a Nueva York.
(c) graficar un polinomio a partir de coeficientes suministrados por el usuario.

32. Escriba un programa en JavaScript que acepte un entero mayor de 2 y que indique si es un número primo. Cabe mencionar que JavaScript tiene instrucciones `if` y `while` con la misma sintaxis que en C y Java. El operador módulo es %. Si necesita la raíz cuadrada de *x*, use *Math.sqrt(x)*.

33. La siguiente es una página de HTML:

```
<html><body>
<a href="www.info-fuente.com/bienvenidos.html">Haga clic para obtener información</a>
</body></html>
```

Si el usuario hace clic en el hipervínculo, se abre una conexión TCP y se envía una serie de líneas al servidor. Liste todas las líneas enviadas.

34. El encabezado *If-Modified-Since* se puede usar para verificar si una página en la caché aún es válida. Se pueden hacer solicitudes de páginas que contengan imágenes, sonido, video, etc., así como HTML. ¿Cree usted que la efectividad de esta técnica es mejor o peor para las imágenes JPEG en comparación con HTML? Piense con cuidado sobre lo que significa "efectividad" y explique su respuesta.

35. En el día de un gran evento deportivo, como el juego de campeonato en algún deporte popular, muchas personas van al sitio web oficial. ¿Es ésta una aglomeración instantánea en el mismo sentido que la elección presidencial en Florida del año 2000? ¿Por qué sí o por qué no?

36. ¿Tiene sentido que un solo ISP funcione como CDN? De ser así, ¿cómo funcionaría? Si no es así, ¿qué tiene de malo la idea?

37. Suponga que no se utiliza compresión para los CD de audio. ¿Cuántos MB de datos debe contener el disco compacto para reproducir dos horas de música?

38. En la figura 7-42(c), el ruido de cuantización ocurre debido a que se usan muestras de 4 bits para representar nueve valores de señal. La primera muestra, en 0, es exacta, pero las siguientes no. ¿Cuál es el error en porcentaje para las muestras en los intervalos 1/32, 2/32 y 3/32 del periodo?

39. ¿Se puede usar un modelo psicoacústico para reducir el ancho de banda necesario para la telefonía de Internet? De ser así, ¿qué condiciones, si las hubiera, habría que satisfacer para que funcionara? De no ser así, ¿por qué no?

40. Un servidor de flujo continuo de audio tiene una “distancia” de un solo sentido de 100 mseg hasta un reproductor de medios. Envía a 1 Mbps. Si el reproductor de medios tiene un búfer de 2 MB, ¿qué puede decir sobre la posición de la marca de nivel bajo y la marca de nivel alto?

41. ¿Tiene la voz sobre IP los mismos problemas con firewalls que el audio de flujo continuo? Explique su respuesta.

42. ¿Cuál es la tasa de bits para transmitir tramas de color de 1200×800 píxeles con 16 bits/píxel a 50 tramas/seg.?

43. ¿Puede un error de 1 bit en una trama MPEG afectar algo más que la trama en la que ocurre el error? Explique su respuesta.

44. Considere un servidor de video con 50 000 clientes, en donde cada cliente ve tres películas por mes. Dos terceras partes de las películas se sirven a las 9 p.m. ¿Cuántas películas tiene que transmitir el servidor a la vez durante este periodo? Si cada película requiere 6 Mbps, ¿cuántas conexiones OC-12 necesita el servidor para la red?

45. Suponga que la ley de Zipf se aplica para los accesos a un servidor de video de 10 000 películas. Si el servidor contiene las 1000 películas más populares en memoria y las 9 000 restantes en disco, proporcione una expresión para la fracción de todas las referencias que se harán a la memoria. Escriba un pequeño programa para evaluar esta expresión en forma numérica.

46. Algunos cyberocupantes (*cybersquatters*) han registrado nombres de dominio que son errores ortográficos de sitios corporativos comunes; por ejemplo, *www.microsfot.com*. Haga una lista de por lo menos cinco de esos dominios.

47. Muchas personas han registrado nombres DNS que consisten en *www.palabra.com*., en donde *palabra* es una palabra común. Para cada una de las siguientes categorías, liste cinco de esos sitios web y sintetice lo que son (por ejemplo, *www.estomago.com* pertenece a un gastroenterólogo en Long Island). He aquí la lista de categorías: animales, alimentos, objetos del hogar y partes del cuerpo. Para la última categoría, por favor apéguese a las partes del cuerpo encima de la cintura.

48. Reescriba el servidor de la figura 6-6 como un servidor web real mediante el uso del comando *GET* para HTTP 1.1. También debe aceptar el mensaje *Host*. El servidor debe mantener una caché de archivos que se hayan obtenido recientemente del disco, y debe atender las solicitudes de la caché cuando sea posible.

8

SEGURIDAD EN REDES

Durante las primeras décadas de su existencia, las redes de computadoras fueron usadas principalmente por investigadores universitarios para el envío de correo electrónico, así como por empleados corporativos para compartir impresoras. En estas condiciones, la seguridad no recibió mucha atención. Pero ahora, cuando millones de ciudadanos comunes usan redes para sus transacciones bancarias, compras y declaraciones de impuestos, y que se ha encontrado una debilidad tras otra, la seguridad de las redes se ha convertido en un problema de proporciones masivas. En este capítulo analizaremos la seguridad de las redes desde varios ángulos, señalaremos muchos peligros y estudiaremos varios algoritmos y protocolos para que sean más seguras.

La seguridad es un tema amplio que cubre una multitud de pecados. En su forma más simple, se ocupa de garantizar que los curiosos no puedan leer, o peor aún, modificar en secreto mensajes dirigidos a otros destinatarios. Tiene que ver con la gente que intenta acceder a servicios remotos no autorizados. También se encarga de mecanismos para verificar que el mensaje supuestamente enviado por la autoridad fiscal que indica: "Pague el viernes o aténgase a las consecuencias" en realidad venga de ella y no de la mafia. La seguridad también se hace cargo del problema de la captura y reproducción de mensajes legítimos, y de las personas que intentan negar que enviaron ciertos mensajes.

La mayoría de los problemas de seguridad son causados intencionalmente por gente maliciosa que intenta ganar algo o hacerle daño a alguien. En la figura 8-1 se muestran algunos de los tipos de transgresores más comunes. Debe quedar claro de esta lista que hacer segura una red comprende mucho más que simplemente mantenerla libre de errores de programación. Implica ser más listo que adversarios a menudo inteligentes, dedicados y a veces bien financiados. Debe quedar claro también que las medidas para detener a los adversarios casuales tendrán poco impacto contra los adversarios serios. Los registros policiales muestran que la mayoría de los ataques no son cometidos por intrusos que interfieren una línea telefónica, sino por miembros internos resentidos. En consecuencia, los sistemas de seguridad deben diseñarse tomando en cuenta este hecho.

Adversario	Objetivo
Estudiante	Divertirse espiando el correo electrónico de otras personas.
Cracker	Probar el sistema de seguridad de alguien; robar datos.
Rep. de ventas	Decir que representa a toda Europa y no sólo a Andorra.
Corporación	Descubrir el plan de marketing estratégico de un competidor.
Ex empleado	Vengarse por haber sido despedido.
Contador	Malversar fondos de una empresa.
Corredor de bolsa	Negar una promesa hecha a un cliente por correo electrónico.
Ladrón de identidad	Robar números de tarjetas de crédito para venderlos.
Gobierno	Conocer los secretos militares o industriales de un enemigo.
Terrorista	Robar secretos de armamento biológico.

Figura 8-1. Algunas personas que podrían causar problemas de seguridad y el porqué de ello.

Los problemas de seguridad de las redes se pueden dividir en términos generales en cuatro áreas interrelacionadas: confidencialidad, autentificación, no repudio y control de integridad. La confidencialidad, también conocida como secrecía, consiste en mantener la información fuera del alcance de usuarios no autorizados. Esto es lo que normalmente viene a la mente cuando la gente piensa en la seguridad de las redes. La autentificación se encarga de determinar con quién se está hablando antes de revelar información delicada o hacer un trato de negocios. El no repudio se encarga de las firmas: ¿cómo comprobar que su cliente en realidad hizo un pedido electrónico por 10 millones de utensilios para zurdos a 89 centavos cada uno, cuando después él afirma que el precio era de 69 centavos? O tal vez argumente que él nunca realizó ningún pedido. Por último, el control de integridad tiene que ver con la forma en que podemos estar seguros de que un mensaje recibido realmente fue el que se envió, y no algo que un adversario malicioso modificó en el camino o ideó por su propia cuenta.

Todos estos temas (confidencialidad, autentificación, no repudio y control de integridad) son pertinentes también en los sistemas tradicionales, pero con algunas diferencias importantes. La confidencialidad y la integridad se logran mediante el uso de correo certificado y al poner los documentos bajo llave. Ahora es más difícil robar el tren del correo de lo que era en la época de Jesse James.

Además, la gente puede por lo general distinguir entre un documento original en papel y una fotocopia, y con frecuencia esto es importante. Como prueba, haga una fotocopia de un cheque válido. Trate de cobrar el cheque original en su banco el lunes. Ahora trate de cobrar la fotocopia del cheque el martes. Observe la diferencia de comportamiento del personal del banco. Con los cheques electrónicos, el original y la copia son idénticos. Es posible que los bancos tarden un poco en acostumbrarse a manejar esto.

La gente autentifica la identidad de otras personas por varios medios; por ejemplo, al reconocer sus caras, voces y letra. Las pruebas de firmas se manejan mediante rúbricas en papel, sellos, etc. Generalmente es posible detectar la alteración de documentos con el auxilio de expertos en escritura, papel y tinta. Ninguna de estas opciones está disponible de forma electrónica. Es obvio que se requieren otras soluciones.

Antes de adentrarnos en las soluciones, vale la pena dedicar un instante a considerar el lugar que corresponde a la seguridad de las redes en la pila de protocolos. Probablemente no haya un solo lugar. Cada capa tiene algo que contribuir. En la capa física podemos protegernos contra la intervención de las líneas de transmisión (o mejor aún, las fibras ópticas) si las encerramos en tubos sellados que contengan un gas inerte a alta presión. Cualquier intento de hacer un agujero en el tubo liberará un poco de gas, con lo cual la presión disminuirá y se disparará una alarma. Algunos sistemas militares usan esta técnica.

En la capa de enlace de datos, los paquetes de una línea punto a punto se pueden encriptar cuando se envíen desde una máquina y desencriptarse cuando lleguen a otra. Todos los detalles se pueden manejar en la capa de enlace de datos, sin necesidad de que las capas superiores se enteren de ello. Sin embargo, esta solución se viene abajo cuando los paquetes tienen que atravesar varios enrutadores, puesto que se tienen que desencriptar en cada enrutador, dentro del cual son vulnerables a posibles ataques. Además, no se contempla que algunas sesiones estén protegidas (por ejemplo, aquellas que involucren compras en línea mediante tarjeta de crédito) y otras no. Sin embargo, la encriptación de enlace (*link encryption*), como se llama a este método, se puede agregar fácilmente a cualquier red y con frecuencia es útil.

En la capa de red se pueden instalar *firewalls* para mantener los paquetes buenos e impedir que entren los paquetes malos. La seguridad de IP también funciona en esta capa.

En la capa de transporte se pueden encriptar conexiones enteras de un extremo a otro; es decir, de proceso a proceso. Para lograr una máxima seguridad, se requiere que ésta sea de extremo a extremo.

Por último, los asuntos como la autentificación de usuario y el no repudio sólo se pueden manejar en la capa de aplicación.

Puesto que la seguridad no encaja por completo en ninguna capa, no se pudo integrar en ningún capítulo de este libro. Por lo tanto, merece su propio capítulo.

Si bien este capítulo es extenso, técnico y esencial, por el momento también es casi irrelevante. Por ejemplo, está bien documentado que la mayoría de las fallas de seguridad en los bancos se deben a procedimientos de seguridad deficientes y a empleados incompetentes, a numerosos errores de implementación que permiten intrusiones remotas por parte de usuarios no autorizados, y a los denominados ataques de ingeniería social, en donde se engaña a los clientes para que revelen los detalles de sus cuentas. Todos estos problemas de seguridad son más frecuentes que los criminales inteligentes que intervienen líneas telefónicas y después decodifican mensajes encriptados. Si una persona pudiera entrar a cualquier sucursal de un banco con una tarjeta de crédito que hubiera encontrado en la calle y solicitara un nuevo NIP argumentando haber olvidado el suyo, y éste se le proporcionara en el acto (en aras de las buenas relaciones con el cliente), toda la criptografía del mundo no podría evitar el abuso. A este respecto, el libro de Ross Anderson (2008a) es una verdadera revelación, debido a que documenta cientos de ejemplos de fallas de seguridad en numerosas industrias, casi todas ellas debidas a lo que cortésmente podríamos llamar prácticas de negocios descuidadas o falta de atención a la seguridad. Sin embargo, pensamos con optimismo que a medida que el comercio electrónico se difunda, el fundamento técnico en el que se base, cuando los factores antes mencionados se cuiden bien, será la criptografía.

Casi toda la seguridad de la red se basa en principios de criptografía, a excepción de la seguridad en la capa física. Por esta razón, para comenzar nuestro estudio de la seguridad examinaremos la criptografía con cierto detalle. En la sección 8.1 veremos algunos de los principios básicos. En las secciones 8.2 a 8.5 examinaremos algunos de los algoritmos y estructuras de datos fundamentales que se utilizan en criptografía. A continuación examinaremos con detalle la forma en que pueden utilizarse estos conceptos para lograr la seguridad en las redes. Concluiremos con algunas breves ideas acerca de la tecnología y la sociedad.

Antes de comenzar, hay que hacer una última aclaración: qué es lo que no se cubre. Hemos tratado de enfocarnos en aspectos de conectividad más que en los de sistema operativo o de aplicación, aunque a menudo es difícil diferenciarlos. Por ejemplo, no hay nada aquí acerca de la autentificación de usuario que utilice biométrica, seguridad mediante contraseña, ataques de desbordamiento de búfer, caballos de Troya, falsificación de inicio de sesión, inyección de código como las secuencias de comandos entre sitios, virus, gusanos y cosas por el estilo. Todos estos temas se cubren con detalle en el capítulo 9 del libro *Sistemas operativos modernos* (Tanenbaum, 2007). El lector interesado en los aspectos de seguridad de los sistemas debe leer ese libro. Ahora comencemos nuestro viaje.

8.1 CRIPTOGRAFÍA

La palabra **criptografía** viene del griego y significa "escritura secreta". Tiene una larga y colorida historia que se remonta a miles de años. En esta sección sólo delinearemos algunos de los puntos de interés, como antecedentes de lo que sigue. Si desea la historia completa de la criptografía, el libro de Kahn (1995) es una de las lecturas recomendadas. Para un tratamiento más completo de la seguridad moderna y los algoritmos criptográficos, los protocolos y las aplicaciones, además del material relacionado, consulte a Kaufman y colaboradores (2002). Para un enfoque más matemático, consulte a Stinson (2002). Para un enfoque menos matemático consulte a Burnett y Paine (2001).

Los profesionales hacen una distinción entre un sistema de cifrado y un sistema de código. Un **sistema de cifrado** es una transformación carácter por carácter o bit por bit, sin importar la estructura lingüística del mensaje. En contraste, uno de **código** reemplaza una palabra con otra palabra o símbolo. Los sistemas de código ya no se utilizan, aunque tienen una historia gloriosa. El código con mayor éxito fue el de las fuerzas armadas de Estados Unidos, durante la Segunda Guerra Mundial en el Pacífico. Tenían indios navajos hablando entre sí, utilizando palabras específicas en navajo correspondientes a términos militares; por ejemplo, *chay-da-gahi-nail-tsaidi* (literalmente: asesino de tortugas) quiere decir arma antitanque. El lenguaje navajo es altamente tonal, muy complejo y no tiene forma escrita. Ninguna persona en Japón sabía algo acerca de él.

En septiembre de 1945, el *San Diego Union* describió el código diciendo: "Durante tres años, en cualquier lugar donde los soldados de la Marina desembarcaban, los japoneses escuchaban una combinación de extraños gorjeos entremezclados con otros ruidos que se asemejaban al canto de un monje tibetano y al sonido de una botella de agua caliente al vaciarse". Los japoneses nunca descifraron el código y muchos de los soldados que utilizaban las claves navajo fueron premiados con altos honores militares por su extraordinario servicio y valor. El hecho de que Estados Unidos descifrara el código japonés y que Japón no haya podido descifrar el navajo, tuvo un papel crucial en las victorias estadounidenses en el Pacífico.

8.1.1 Introducción a la criptografía

Históricamente, cuatro grupos han utilizado y contribuido al arte de la criptografía: los militares, el cuerpo diplomático, los redactores de los periódicos y los amantes. De éstos, la milicia ha tenido el papel más importante y ha moldeado el campo a través de los siglos. Dentro de las organizaciones militares, los mensajes por encriptar se han entregado de forma tradicional a empleados mal pagados para su cifrado y transmisión. El gran volumen de mensajes ha provocado asignar esta labor a unos cuantos especialistas.

Hasta la llegada de las computadoras, una de las principales restricciones de la criptografía había sido la capacidad del empleado encargado de la codificación para realizar las transformaciones necesarias, con frecuencia en un campo de batalla con poco equipo. Una restricción adicional ha sido la dificultad de cambiar rápidamente de un método de criptografía a otro, debido a que esto implica volver a capacitar a una gran cantidad de personas. Sin embargo, el peligro de que un empleado fuera capturado por el enemigo ha hecho indispensable la capacidad de cambiar el método de criptografía de manera instantánea, de ser necesario. Estos requerimientos en conflicto han dado lugar al modelo de la figura 8-2.

Los mensajes a encriptar, conocidos como **texto plano**, se transforman mediante una función parametrizada por una clave. El resultado del proceso de encriptación, conocido como **texto cifrado**, se transmite a continuación, muchas veces por mensajero o radio. Suponemos que el enemigo, o **intruso**, escucha y copia con exactitud todo el texto cifrado. Sin embargo, a diferencia del destinatario original, el intruso no conoce la clave de desencriptación y no puede desifrar con facilidad este texto. En ocasiones no sólo puede escuchar el canal de comunicación (intruso pasivo), sino también puede registrar mensajes

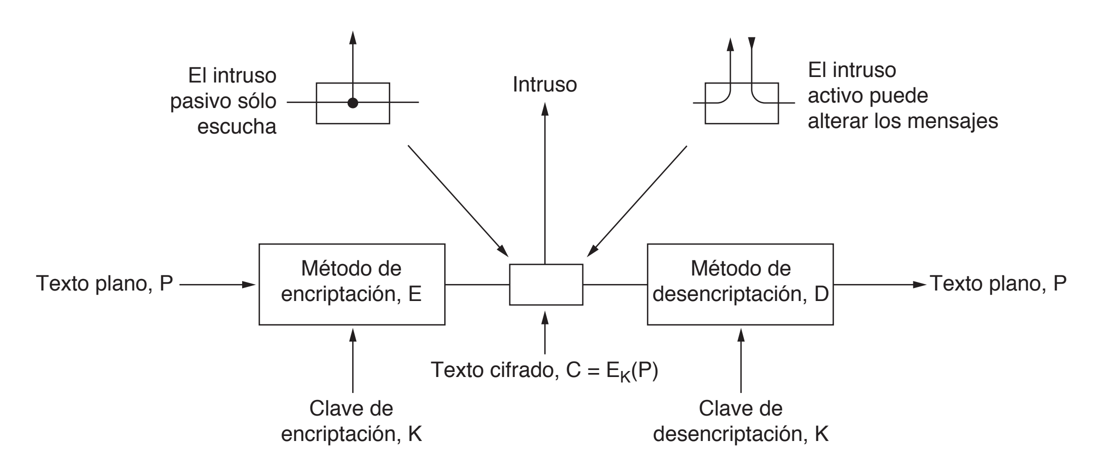


Figura 8-2. Modelo de encriptación (para un sistema de cifrado de clave simétrica).

y posteriormente reproducirlos, inyectar sus propios mensajes o modificar los mensajes legítimos antes de que lleguen al receptor (intruso activo). El arte de quebrantar los sistemas de cifrado, es conocido como **criptoanálisis**, y en conjunto con el arte de idearlos (criptografía) se conocen como **criptología**.

A menudo resulta útil tener una notación para relacionar el texto plano, el texto cifrado y las claves. Utilizaremos $C = E_K(P)$ para indicar que la encriptación del texto plano P mediante el uso de la clave K produce el texto cifrado C. Del mismo modo, $P = D_K(C)$ representa la desencriptación de C para obtener el texto plano una vez más. Por lo tanto,

$$D_K(E_K(P)) = P$$

Esta notación sugiere que E y D son sólo funciones matemáticas, lo cual es cierto. El único truco es que ambas son funciones de dos parámetros, y hemos escrito uno de los parámetros (la clave) como subíndice, en lugar de como argumento, para distinguirlo del mensaje.

Una regla fundamental de la criptografía es que se debe suponer que el criptoanalista conoce los métodos utilizados para encriptar y desencriptar. En otras palabras, el criptoanalista sabe con detalle cómo funciona el método de encriptación, E, y el de desencriptación, D, de la figura 8-2. La cantidad de esfuerzo necesario para inventar, probar e instalar un nuevo algoritmo cada vez que el método antiguo está en peligro (o se piensa que lo está) siempre ha hecho poco práctico mantener en secreto el algoritmo de encriptación. Además, pensar que dicho algoritmo es secreto cuando no lo es, hace más daño que bien.

Aquí es donde entra la clave. Ésta consiste en una cadena corta (relativamente) que selecciona uno de muchos encriptados potenciales. En contraste con el método general, que tal vez sólo se cambie cada cierto número de años, la clave se puede cambiar con la frecuencia requerida. Por lo tanto, nuestro modelo básico es un método general estable y conocido públicamente, pero parametrizado por una clave secreta y que puede cambiarse con facilidad. La idea de que el criptoanalista conozca los algoritmos y que la naturaleza secreta se base principalmente en las claves se conoce como **Principio de Kerckhoff**, que debe su nombre al criptógrafo militar holandés Auguste Kerckhoff, quien lo estableció por primera vez en 1883 (Kerckhoff, 1883). Por lo tanto, tenemos:

Principio de Kerckhoff: Todos los algoritmos deben ser públicos; sólo las claves deben ser secretas.

La naturaleza no secreta del algoritmo no puede enfatizarse lo suficiente. Tratar de mantener oculto el algoritmo, lo que se conoce como **seguridad por desconocimiento**, nunca funciona. Además, al hacer público el algoritmo, el criptógrafo recibe asesoría gratuita de una gran cantidad de criptólogos académicos ansiosos por descifrar el sistema, con el propósito de publicar trabajos que demuestren su inteligencia.

Si muchos expertos han tratado de descifrar el algoritmo durante mucho tiempo después de su publicación y nadie lo ha logrado, probablemente sea bastante sólido.

Puesto que la parte secreta debe ser la clave, la longitud de ésta es un aspecto importante del diseño. Considere una simple cerradura de combinación. El principio general es que se introducen dígitos en secuencia. Todo el mundo lo sabe, pero la clave es secreta. Una longitud de clave de dos dígitos significa que hay 100 posibilidades. Una clave de tres dígitos significa que hay 1000 posibilidades y una clave de seis dígitos de longitud significa un millón. Cuanto más grande sea la clave, mayor será el factor de trabajo que tendrá que enfrentar el criptoanalista. El **factor de trabajo** para descifrar el sistema mediante una búsqueda exhaustiva del espacio de clave crece exponencialmente con la longitud de la clave. El secreto radica en tener un algoritmo poderoso (pero público) y una clave larga. Para evitar que su hermano menor lea su correo electrónico, las claves de 64 bits son suficientes. Para el uso comercial común, se requieren por lo menos 128 bits. Para mantener a raya a gobiernos poderosos se requieren claves de al menos 256 bits, y de preferencia mayores.

Desde el punto de vista del criptoanalista, el problema del criptoanálisis presenta tres variaciones principales. Cuando el criptoanalista cuenta con cierta cantidad de texto cifrado pero no tiene texto plano, se enfrenta al problema de **sólo texto cifrado**. Los criptogramas que aparecen en la sección de acertijos de los diarios presentan este tipo de problema. Cuando el criptoanalista cuenta con texto cifrado y el texto plano correspondiente, se enfrenta al problema llamado **texto plano conocido**. Por último, cuando el criptoanalista tiene la capacidad de encriptar piezas de texto plano de su propia elección, tenemos el problema del **texto plano seleccionado**. Los criptogramas de los periódicos se podrían descifrar con facilidad si el criptoanalista pudiera hacer preguntas como: "¿cuál es el encriptado de ABCDEFGHIJKL?"

Los principiantes en el campo de la criptografía con frecuencia suponen que, si un sistema de cifrado puede resistir el ataque de sólo texto cifrado, entonces es seguro. Esta suposición es muy ingenua. En muchos casos, el criptoanalista puede hacer una buena estimación de partes del texto plano. Por ejemplo, lo primero que muchas computadoras dicen cuando se les llama es "login:", o "Iniciar sesión:". Equipado con algunos pares concordantes de texto plano-texto cifrado, el trabajo del criptoanalista se vuelve mucho más fácil. Para lograr seguridad, el criptógrafo debe ser conservador y asegurarse de que el sistema sea inviolable aun si su oponente puede encriptar cantidades arbitrarias de texto plano seleccionado.

Los métodos de encriptación se han dividido históricamente en dos categorías: sistema de cifrado por sustitución y sistema de cifrado por transposición. A continuación analizaremos cada uno de éstos como antecedente para la criptografía moderna.

8.1.2 Sistemas de cifrado por sustitución

En un **sistema de cifrado por sustitución**, cada letra o grupo de letras se reemplazan por otra letra o grupo de letras para disfrazarla. Uno de los sistemas de cifrado más viejos conocido es el **sistema de cifrado de César**, atribuido a Julio César. En este método, *a* se vuelve *D*, *b* se vuelve *E*, *c* se vuelve *F*, ... , y *z* se vuelve *C*. Por ejemplo, *ataque* se vuelve *DWDTXH*. En nuestros ejemplos, el texto plano se presentará en minúsculas y el texto cifrado en mayúsculas.

Una ligera generalización del sistema de cifrado de César permite que el alfabeto de texto cifrado se desplace *k* letras, en lugar de siempre tres. En este caso, *k* se convierte en una clave del método general de alfabetos desplazados circularmente. Quizás el sistema de cifrado de César haya engañado a Pompeyo, pero no ha engañado a nadie más desde entonces.

La siguiente mejora es hacer que cada uno de los símbolos del texto plano, digamos las 26 letras del abecedario inglés (no incluimos la ñ), se asocien con alguna otra letra. Por ejemplo,

```
texto plano:    a b c d e f g h i j k l m n o p q r s t u v w x y z
texto cifrado:  Q W E R T Y U I O P A S D F G H J K L Z X C V B N M
```

El sistema general de sustitución de símbolo por símbolo se llama sistema de **cifrado por sustitución monoalfabética**, siendo la clave la cadena de 26 letras correspondiente al alfabeto completo. Para la clave anterior, el texto plano *ataque* se transformaría en el texto cifrado *QZQJXT*.

A primera vista, esto podría parecer un sistema seguro porque, aunque el criptoanalista conoce el sistema general (sustitución letra por letra), no sabe cuál de las $26! \approx 4 \times 10^{26}$ claves posibles se está usando. En contraste con el sistema de cifrado de César, intentarlas todas no es una estrategia muy alentadora. Incluso a 1 nseg por solución, un millón de chips de computadora trabajando en paralelo tardarían 10 000 años en probar todas las claves.

No obstante, si se cuenta con una cantidad muy pequeña de texto cifrado, puede descifrarse fácilmente. El ataque básico aprovecha las propiedades estadísticas de los lenguajes naturales. En inglés, por ejemplo, la *e* es la letra más común, seguida de *t*, *o*, *a*, *n*, *i*, etc. Las combinaciones de dos letras más comunes, o **digramas**, son *th*, *in*, *er*, *re* y *an*. Las combinaciones de tres letras más comunes, o **trigramas**, son *the*, *ing*, *and* e *ion*.

Un criptoanalista que intentara descifrar un sistema de cifrado monoalfabético comenzaría por contar las frecuencias relativas de todas las letras en el texto cifrado. Entonces podría asignar tentativamente la más común a la letra *e* y la siguiente más común a la letra *t*. Vería entonces los trigramas para encontrar uno común de la forma *tXe*, lo que sugerirá fuertemente que *X* es *h*. De la misma manera, si el patrón *thYt* ocurre con frecuencia, *Y* quizá representa la letra *a*. Con esta información el criptoanalista puede buscar un trigrama frecuente de la forma *aZW*, y lo más probable es que sea *and*. Al adivinar las palabras comunes, digramas y trigramas, y al conocer los patrones probables de las vocales y consonantes, el criptoanalista construye un texto plano tentativo, letra por letra.

Otra estrategia es adivinar una palabra o frase probable. Por ejemplo, considere el siguiente texto cifrado de un despacho contable (en bloques de cinco caracteres):

```
CTBMN  BYCTC  BTJDS  QXBNS  GSTJC  BTSWX  CTQTZ  CQVUJ
QJSGS  TJQZZ  MNQJS  VLNSX  VSZJU  JDSTS  JQUUS  JUBXJ
DSKSU  JSNTK  BGAQJ  ZBGYQ  TLCTZ  BNYBN  QJSW
```

Una palabra muy probable en un mensaje de un despacho contable de un país de habla inglesa es *financial*. Puesto que sabemos que *financial* tiene una letra repetida (*i*), con otras cuatro letras entre sus ocurrencias, buscamos letras repetidas en el texto cifrado con este espaciado. Encontramos 12 casos, en las posiciones 6, 15, 27, 31, 42, 48, 56, 66, 70, 71, 76 y 82. Sin embargo, sólo dos de éstos, el 31 y 42, tienen la siguiente letra (correspondiente a *n* en el texto plano) repetida en el lugar adecuado. De estos dos, sólo el 31 tiene también la *a* en la posición correcta, por lo que sabemos que *financial* comienza en la posición 30. A partir de aquí, es fácil deducir la clave mediante el uso de las estadísticas de frecuencia del texto en inglés y al buscar palabras casi completas para terminarlas.

8.1.3 Sistemas de cifrado por transposición

Los sistemas de cifrado por sustitución conservan el orden de los símbolos de texto plano, pero los disfrazan. En contraste, los sistemas de cifrado por transposición reordenan las letras pero no las disfrazan. En la figura 8-3 se presenta un **sistema de cifrado por transposición** común, la transposición columnar. La clave del sistema de cifrado es una palabra o frase que no contiene letras repetidas. En este ejemplo, la clave es MEGABUCK. El propósito de la clave es ordenar las columnas; la columna 1 está bajo la letra clave más cercana al inicio del alfabeto, y así sucesivamente. El texto plano se escribe de manera horizontal, en filas, las cuales se rellenan para completar la matriz si es necesario. El texto cifrado se lee por columnas, comenzando por la columna cuya letra clave es la más baja.

M	E	G	A	B	U	C	K
7	4	5	1	2	8	3	6
p	l	e	a	s	e	t	r
a	n	s	f	e	r	o	n
e	m	i	l	l	i	o	n
d	o	l	l	a	r	s	t
o	m	y	s	w	i	s	s
b	a	n	k	a	c	c	o
u	n	t	s	i	x	t	w
o	t	w	o	a	b	c	d

Texto plano
pleasetransferonemilliondollarsto
myswissbankaccountsixtwotwo

Texto cifrado
AFLLSKSOSELAWAIATOOSSCTCLNMOMANT
ESILYNTWRNNTSOWDPAEDOBUOERIRICXB

Figura 8-3. Un sistema de cifrado por transposición.

Para descifrar un sistema de cifrado por transposición, el criptoanalista debe primero estar consciente de que está tratando con un sistema de este tipo. Al observar la frecuencia de *E*, *T*, *A*, *O*, *I*, *N*, etc., es fácil ver si se ajustan al patrón usual del texto plano. De ser así, es evidente que se trata de un sistema de cifrado por transposición, pues en tal sistema de cifrado cada letra se representa a sí misma y la distribución de frecuencia permanece intacta.

El siguiente paso es adivinar la cantidad de columnas. En muchos casos es posible adivinar una palabra o frase probable mediante el contexto del mensaje. Por ejemplo, suponga que nuestro criptoanalista sospecha que la frase de texto plano *milliondollars* aparece en algún lugar del mensaje. Observe que los digramas *MO*, *IL*, *LL*, *LA*, *IR* y *OS* aparecen en el texto cifrado como resultado de la envoltura de la frase. La letra de texto cifrado *O* sigue a la letra de texto cifrado *M* (es decir, son verticalmente adyacentes en la columna 4) puesto que están separados en la frase probable por una distancia igual a la longitud de la clave. Si se hubiera usado una clave de longitud siete, habrían aparecido los digramas *MD*, *IO*, *LL*, *LL*, *IA*, *OR* y *NS*. De hecho, para cada longitud de clave, se produce un grupo diferente de digramas en el texto cifrado. Al buscar las distintas posibilidades, el criptoanalista puede determinar con frecuencia la longitud de la clave.

El último paso es ordenar las columnas. Cuando la cantidad de columnas k es pequeña, puede examinarse cada uno de los pares de columnas $k(k-1)$ para ver si las frecuencias de sus digramas coinciden con las del texto plano en inglés. Se asume que el par con la mejor coincidencia está ubicado de manera correcta. Ahora cada una de las columnas restantes se prueba tentativamente como sucesora de este par. La columna cuyas frecuencias de digramas y trigramas producen la mejor coincidencia se toma tentativamente como correcta. La siguiente columna se encuentra de la misma manera. El proceso completo se repite hasta encontrar un orden potencial. Es probable que en este punto se pueda reconocer texto plano (por ejemplo, si aparece *milloin*, quedará claro en dónde está el error).

Algunos sistemas de cifrado por transposición aceptan un bloque de longitud fija como entrada y producen un bloque de longitud fija como salida. Para describir estos sistemas de cifrado por completo hay que proporcionar una lista que indique el orden en el que deben salir los caracteres. Por ejemplo, el sistema de cifrado de la figura 8-3 se puede ver como un sistema de cifrado de bloque de 64 caracteres. Su salida es 4, 12, 20, 28, 36, 44, 52, 60, 5, 13, . . . , 62. En otras palabras, el cuarto carácter de entrada, *a*, es el primero en salir, seguido del decimosegundo, *f*, y así en lo sucesivo.

8.1.4 Rellenos de una sola vez

En realidad es bastante sencillo construir un sistema de cifrado inquebrantable; la técnica se conoce desde hace décadas. Primero se escoge una cadena de bits al azar como clave. Luego se convierte el texto plano en una cadena de bits; por ejemplo, mediante el uso de su representación ASCII. Por último, se calcula el XOR (OR exclusivo) de estas dos cadenas, bit por bit. El texto cifrado resultante no se puede descifrar,

porque en una muestra suficientemente grande de texto cifrado cada letra aparecerá con la misma frecuencia, lo mismo que cada diagrama, trigrama, y así sucesivamente. Este método, conocido como **relleno de una sola vez**, es inmune a todos los ataques actuales y futuros sin importar cuánta potencia computacional tenga el intruso. La razón se deriva de la teoría de la información: simplemente no hay información en el mensaje debido a que todos los textos planos posibles de una longitud dada son parecidos.

En la figura 8-4 se muestra un ejemplo de cómo se utilizan los rellenos de una sola vez. Primero, el mensaje 1, "I love you." se convierte en código ASCII de 7 bits. A continuación se elige el relleno de una sola vez, relleno 1, y se le aplica un XOR con el mensaje para obtener el texto cifrado. Un criptoanalista podría probar todos los posibles rellenos de una sola vez para ver qué texto plano proviene de cada uno. Por ejemplo, podría probarse el relleno 2 de la figura, lo que resultaría en el texto plano 2, "Elvis lives", lo cual podría ser o no verosímil (un tema que está fuera del alcance de este libro). De hecho, para cada texto plano ASCII de 11 caracteres hay un relleno de una sola vez que lo genera. Eso es lo que queremos dar a entender cuando decimos que no hay información en el texto cifrado: es posible obtener cualquier mensaje con la longitud correcta a partir de él.

Mensaje 1:	1001001	0100000	1101100	1101111	1110110	1100101	0100000	1111001	1101111	1110101	0101110
Relleno 1:	1010010	1001011	1110010	1010101	1010010	1100011	0001011	0101010	1010111	1100110	0101011
Texto cifrado:	0011011	1101011	0011110	0111010	0100100	0000110	0101011	1010011	0111000	0010011	0000101
Relleno 2:	1011110	0000111	1101000	1010011	1010111	0100110	1000111	0111010	1001110	1110110	1110110
Texto plano 2:	1000101	1101100	1110110	1101001	1110011	0100000	1101100	1101001	1110110	1100101	1110011

Figura 8-4. El uso de un relleno de una sola vez para encriptación y la posibilidad de obtener cualquier texto plano posible a partir del texto cifrado mediante el uso de un relleno diferente.

En teoría, los rellenos de una sola vez son excelentes, pero en la práctica tienen varias desventajas. Para comenzar, la clave no se puede memorizar, por lo que tanto el emisor como el receptor deben transportar una copia escrita con ellos. Si cualquiera de ellos corre el peligro de ser capturado, es obvio que las claves escritas no son una buena idea. Además, la cantidad total de datos que se pueden transmitir está limitada por la cantidad de claves disponibles. Si el espía tiene suerte y descubre una gran cantidad de datos, quizá no los pueda transmitir al cuartel general debido a que la clave se ha agotado. Otro problema es la sensibilidad del método a los caracteres perdidos o insertados. Si el emisor y el receptor pierden la sincronización, de ahí en adelante todos los datos aparecerán distorsionados.

Con la aparición de las computadoras, el relleno de una sola vez podría volverse práctico para algunas aplicaciones. El origen de la clave podría ser un DVD especial que contenga varios gigabytes de información y, si se transporta en una caja de película para DVD y se le anteponen algunos minutos de video, ni siquiera sería sospechoso. Por supuesto que, a velocidades de red de gigabit, tener que insertar un nuevo DVD cada 30 segundos podría volverse tedioso. Y los DVD deben transportarse personalmente desde el emisor hasta el receptor antes de poder enviar cualquier mensaje, lo que reduce en gran medida su utilidad práctica.

Criptografía cuántica

Curiosamente, podría haber una solución al problema de cómo transmitir el relleno de una sola vez a través de la red, y proviene de un origen muy inverosímil: la mecánica cuántica. Esta área aún es experimental, pero las pruebas iniciales son alentadoras. Si pudiera perfeccionarse y fuera eficiente, con el tiempo casi toda la criptografía se realizaría utilizando rellenos de una sola vez debido a su seguridad comprobada. A continuación explicaremos brevemente cómo funciona el método de la **criptografía cuántica**.

En particular describiremos un protocolo llamado **BB84**, que debe su nombre a sus autores y a su año de publicación (Bennet y Brassard, 1984).

Suponga que un usuario, de nombre Alice, desea establecer un relleno de una sola vez con un segundo usuario, Bob. Alice y Bob son los protagonistas, los personajes principales en nuestra historia. Por ejemplo, Bob es un banquero con quien Alice quiere hacer negocios. Los nombres "Alice" y "Bob" se han utilizado como protagonistas en casi todo lo escrito sobre criptografía desde que Ron Rivest los presentara hace muchos años atrás (Rivest y colaboradores, 1978). Los criptógrafos aman la tradición. Si utilizáramos "Andy" y "Bárbara" como protagonistas, nadie creería nada de lo que se dice en este capítulo. Por lo tanto, dejémoslo así.

Si Alice y Bob pudieran establecer un relleno de una sola vez, podrían utilizarlo para comunicarse de manera segura. La pregunta es: ¿cómo pueden establecerlo sin intercambiar antes algunos DVD? Podemos asumir que se encuentran en extremos opuestos de un cable de fibra óptica a través del cual pueden enviar y recibir pulsos de luz. Sin embargo, un intruso intrépido, Trudy, puede cortar la fibra para establecer una derivación activa. Trudy puede leer todos los bits en ambas direcciones. También puede enviar mensajes falsos en ambas direcciones. Para Alice y Bob la situación podría parecer irremediable, pero la criptografía cuántica puede arrojarle un poco de luz al tema en cuestión.

La criptografía cuántica se basa en el hecho de que la luz viene en pequeños paquetes llamados **fotones**, los cuales tienen algunas propiedades peculiares. Además, la luz se puede polarizar al pasarla a través de un filtro de polarización, un hecho bien conocido por quienes utilizan anteojos oscuros y por los fotógrafos. Si se pasa un haz de luz (es decir, un flujo de fotones) a través de un filtro polarizador, todos los fotones que emerjan de él se polarizarán en la dirección del eje del filtro (por ejemplo, el vertical). Si a continuación el haz se pasa a través de un segundo filtro polarizador, la intensidad de la luz que emerja del segundo filtro es proporcional al cuadrado del coseno del ángulo entre los ejes. Si los dos ejes son perpendiculares, no pasa ningún fotón. La orientación absoluta de los dos filtros no importa; sólo cuenta el ángulo entre sus ejes.

Para generar un relleno de una sola vez, Alice necesita dos conjuntos de filtros polarizadores. El primer conjunto consiste en un filtro vertical y en uno horizontal. Esta opción se conoce como **base rectilínea**. Una base es en esencia un sistema de coordenadas. El segundo conjunto de filtros consiste en lo mismo, excepto que se rotan 45 grados, de forma que un filtro va del extremo inferior izquierdo al extremo superior derecho y el otro va del extremo superior izquierdo al extremo inferior derecho. Esta opción se conoce como **base diagonal**. Por lo tanto, Alice tiene dos bases, las cuales puede insertar a voluntad y rápidamente en su haz. En realidad, Alice no tiene cuatro filtros separados, sino un cristal cuya polarización se puede cambiar de manera eléctrica y a gran velocidad a cualquiera de las cuatro direcciones permitidas. Bob tiene el mismo equipo que Alice. El hecho de que Alice y Bob dispongan de dos bases cada uno es esencial para la criptografía cuántica.

Ahora Alice asigna una dirección como 0 y la otra como 1 para cada base. En el ejemplo siguiente suponemos que elige 0 para vertical y 1 para horizontal. De manera independiente, también elige 0 para la dirección del extremo inferior izquierdo al superior derecho y 1 para la dirección del extremo superior izquierdo al inferior derecho. Alice envía estas opciones a Bob como texto plano.

A continuación, Alice elige un relleno de una sola vez; por ejemplo, con base en un generador de números aleatorios (un tema complejo por sí solo). Lo transmite bit por bit a Bob y elige de manera aleatoria una de sus dos bases para cada bit. Para enviar un bit, su pistola de fotones emite un fotón polarizado de manera apropiada para la base que está utilizando para ese bit. Por ejemplo, podría elegir bases diagonal, rectilínea, rectilínea, diagonal, rectilínea, etc. Para enviarle un relleno de una sola vez de 1001110010100110 con estas bases, Alice debería enviar los fotones que se muestran en la figura 8-5(a). Dados el relleno de una sola vez y la secuencia de bases, la polarización por utilizar para cada bit se determina de manera única. Los bits enviados en un fotón a la vez se conocen como **qubits**.

Bob no sabe cuáles bases utilizar, por lo que elige una al azar para cada fotón entrante y simplemente la usa, como se muestra en la figura 8-5(b). Si elige la base correcta, obtiene el bit correcto. Si elige la base incorrecta, obtiene un bit aleatorio debido a que si un fotón choca con un filtro que está polarizado a 45 grados de su propia polarización, salta de manera aleatoria a la polarización del filtro o a una polarización perpendicular al filtro con igual probabilidad. Esta propiedad de los fotones es fundamental para la mecánica cuántica. Por lo tanto, algunos de los bits son correctos y algunos son aleatorios, pero Bob no sabe cuál es cuál. Los resultados de Bob se muestran en la figura 8-5(c).

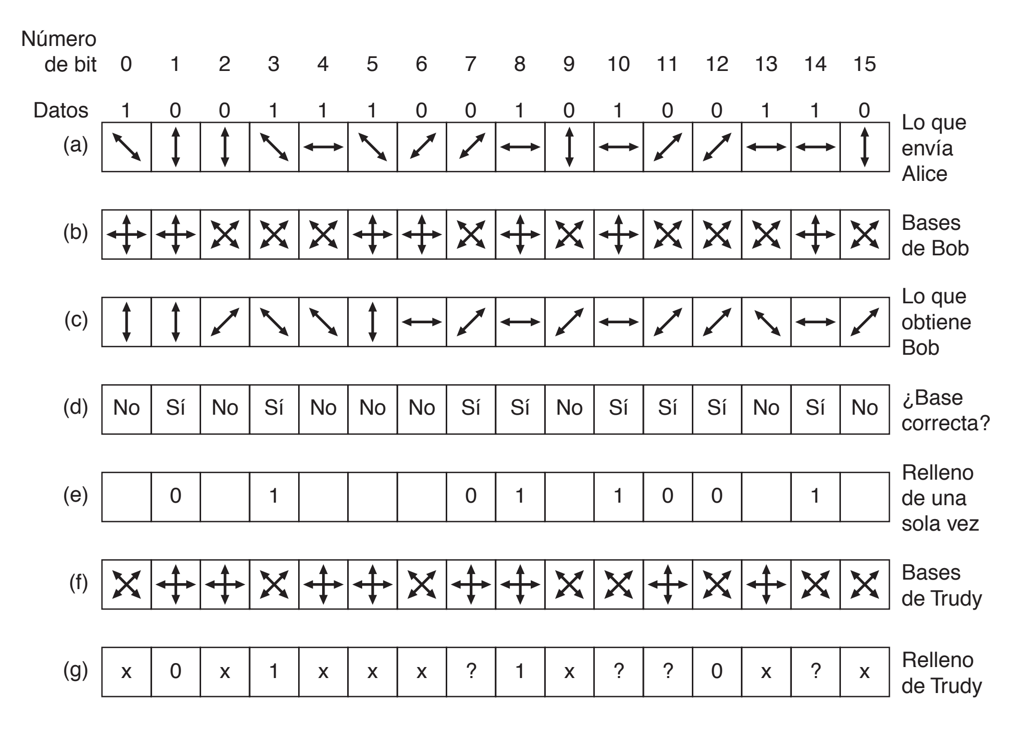


Figura 8-5. Un ejemplo de criptografía cuántica.

¿Cómo sabe Bob cuáles de las bases que obtuvo son correctas y cuáles erróneas? Simplemente indica a Alice, mediante texto plano, cuál base utilizó para cada bit y ella le indica a él, también en texto plano, cuáles son correctas y cuáles erróneas, como se muestra en la figura 8-5(d). A partir de esta información los dos pueden construir una cadena de bits tomando en cuenta los aciertos, como se muestra en la figura 8-5(e). En promedio, esta cadena de bits tendrá la mitad de la longitud de la cadena de bits original, pero debido a que las dos partes lo saben, ambas pueden utilizarla como un relleno de una sola vez. Todo lo que Alice tiene que hacer es transmitir una cadena de bits un poco mayor que el doble de la longitud deseada, y tanto ella como Bob tendrán un relleno de una sola vez con la longitud deseada. Problema resuelto.

Pero espere un momento. Nos olvidamos de Trudy. Suponga que ella tiene curiosidad por saber lo que dirá Alice, así que corta la fibra e inserta sus propios detector y transmisor. Por desgracia para ella, tampoco sabe cuál base utilizar para cada fotón. Lo mejor que puede hacer es elegir una al azar para cada fotón, tal como lo hizo Bob. En la figura 8-5(f) se muestra un ejemplo de sus elecciones. Cuando más tarde Bob informó (en texto plano) cuáles bases utilizó y Alice le indica (en texto plano) cuáles fueron correctas, Trudy sabe cuándo acertó y cuándo no. En la figura 8-5 acertó en los bits 0, 1, 2, 3, 4, 6, 8, 12 y 13. Pero sabe a partir de la respuesta de Alice en la figura 8-5(d) que sólo los bits 1, 3, 7, 8, 10, 11, 12 y 14 son parte del

relleno de una sola vez. Acertó y capturó el bit correcto en cuatro de estos bits (1, 3, 8 y 12). En los otros cuatro (7, 10, 11 y 14) no adivinó y no conoce el bit transmitido. Por lo tanto, a partir de la figura 8-5(e), Bob sabe que el relleno de una sola vez inicia con 01011001, pero todo lo que Trudy tiene es 01?1??0?, tomando en cuenta la figura 8-5(g).

Por supuesto que Alice y Bob están conscientes de que tal vez Trudy haya capturado parte de su relleno de una sola vez, por lo que les gustaría reducir la información que ésta tiene. Para ello pueden realizar una transformación sobre el relleno. Por ejemplo, pueden dividir el relleno de una sola vez en bloques de 1024 bits, elevar al cuadrado cada uno de ellos para formar un número de 2048 bits y utilizar la concatenación de estos números de 2 048 bits como el relleno de una sola vez. Con su conocimiento parcial de la cadena de bits transmitida, Trudy no tiene forma de elevar al cuadrado y, por lo tanto, no tiene nada. La transformación del relleno original de una sola vez a uno diferente que reduce el conocimiento de Trudy se conoce como **amplificación de privacidad**. En la práctica, en lugar de las elevaciones al cuadrado se utilizan transformaciones complejas en las que cada bit de salida depende de cada bit de entrada.

Pobre Trudy. No sólo no tiene idea de cuál es el relleno de una sola vez, sino que su presencia tampoco es un secreto. Después de todo, debe retransmitir a Bob cada bit recibido para hacerle creer que está hablando con Alice. El problema es que lo mejor que puede hacer es transmitir el qubit que recibió, utilizando la polarización que utilizó para recibirlo, y casi la mitad del tiempo estará mal, lo cual provocaría muchos errores en el relleno de una sola vez de Bob.

Cuando Alice finalmente comienza a enviar datos, los codifica mediante un pesado código de corrección de errores hacia adelante. Desde el punto de vista de Bob, un error de un 1 bit en el relleno de una sola vez es lo mismo que un error de transmisión de 1 bit. De cualquier forma, obtiene el bit incorrecto. Si hay suficiente corrección de errores hacia adelante, puede recuperar el mensaje original a pesar de todos los errores, y puede contar fácilmente cuántos errores fueron corregidos. Si este número es mayor que la tasa de errores esperada del equipo, él sabe que Trudy ha intervenido la línea y puede actuar en consecuencia (por ejemplo, decirle a Alice que cambie a un canal de radio, que llame a la policía, etc.). Si Trudy tuviera alguna forma de clonar un fotón con el fin de contar con un fotón para inspeccionar y uno idéntico para enviar a Bob, podría evitar la detección, pero en la actualidad no se conoce ninguna forma de clonar perfectamente un fotón. E incluso aunque Trudy pudiera clonar fotones, no se podría reducir el valor de la criptografía cuántica para establecer rellenos de una sola vez.

Aunque se ha mostrado que la criptografía cuántica puede operar a través de distancias de 60 km de fibra, el equipo es complejo y costoso. Aun así, la idea es prometedora. Para obtener mayor información acerca de la criptografía cuántica, consulte a Mullins (2002).

8.1.5 Dos principios criptográficos fundamentales

Aunque estudiaremos muchos sistemas criptográficos diferentes en las siguientes páginas, hay dos principios que los sostienen a todos y que su comprensión es importante. Ponga atención. Si los quebranta, será bajo su riesgo.

Redundancia

El primer principio es que todos los mensajes encriptados deben contener redundancia; es decir, información no necesaria para entender el mensaje. Un ejemplo puede dejar en claro la necesidad de esto. Considere una compañía de compras por correo, El Perezoso (EP), que tiene 60 000 productos. Pensando que son muy eficientes, los programadores de EP deciden que los mensajes de pedidos deben consistir en un nombre de cliente de 16 bytes seguido de un campo de datos de 3 bytes (1 byte para la cantidad y 2 para el número de producto). Los últimos 3 bytes se deben encriptar mediante el uso de una clave muy extensa, conocida sólo por el cliente y EP.

En un principio, esto podría parecer seguro, y en cierto sentido lo es, puesto que los intrusos pasivos no pueden desencriptar los mensajes. Por desgracia, el sistema también tiene una falla mortal que lo vuelve inútil. Supongamos que una empleada recientemente despedida quiere vengarse de EP por haberla cesado. Antes de irse, se lleva con ella la lista de clientes. Después, trabaja toda la noche escribiendo un programa para generar pedidos ficticios usando nombres reales de los clientes. Dado que no tiene la lista de claves, sólo pone números aleatorios en los últimos 3 bytes y envía cientos de pedidos a EP.

Al llegar estos mensajes, la computadora de EP usa el nombre del cliente para localizar la clave y desencriptar el mensaje. Por desgracia para EP, casi todos los mensajes de 3 bytes son válidos, por lo que la computadora comienza a imprimir instrucciones de envío. Aunque podría parecer extraño que un cliente ordene 837 juegos de columpios para niños o 540 cajas de arena, en lo que a la computadora concierne el cliente bien podría estar planeando abrir una cadena de franquicias con juegos infantiles. De esta manera, un intruso activo (la ex empleada) puede causar muchísimos problemas, aun cuando no pueda entender los mensajes que genera su computadora.

Podemos resolver este problema si agregamos redundancia a todos los mensajes. Por ejemplo, si se extienden los mensajes de pedido a 12 bytes, de los cuales los primeros 9 deben ser ceros, entonces este ataque ya no funciona, porque el ex empleado ya no puede generar una larga cadena de mensajes válidos. La moraleja de esta historia es que todos los mensajes deben contener una cantidad considerable de redundancia, para que los intrusos activos no puedan enviar basura al azar y lograr que se interprete como mensajes válidos.

Sin embargo, la adición de redundancia también simplifica a los criptoanalistas el descifrado de los mensajes. Suponga que el negocio de pedidos por correo es altamente competitivo, y que la competencia principal de El Perezoso, El Bolsón, estaría encantado de conocer la cantidad de cajas de arena que EP vende, para lo cual interviene la línea telefónica de EP. En el esquema original con mensajes de 3 bytes, el criptoanálisis era prácticamente imposible puesto que, tras adivinar una clave, el criptoanalista no tenía manera de saber si había adivinado correctamente, ya que casi todos los mensajes eran técnicamente legales. Con el nuevo esquema de 12 bytes, es fácil para el criptoanalista distinguir un mensaje válido de uno no válido. Por lo tanto, tenemos:

Principio criptográfico 1: Los mensajes deben contener alguna redundancia.

En otras palabras, al desencriptar un mensaje, el destinatario debe tener la capacidad de saber si es válido con sólo inspeccionarlo y tal vez mediante un cálculo simple. Esta redundancia es necesaria para evitar que intrusos activos envíen basura y engañen al receptor para que desencripte la basura y realice algo con el "texto plano". Sin embargo, esta misma redundancia simplifica en gran medida la violación del sistema por parte de los intrusos pasivos, por lo que aquí hay un poco de preocupación. Además, la redundancia nunca debe estar en la forma de n ceros al principio o al final de un mensaje, debido a que al ejecutar tales mensajes a través de algunos algoritmos criptográficos los resultados son más predecibles, lo que facilita el trabajo del criptoanalista. Un polinomio CRC es mucho mejor que una serie de ceros, puesto que el receptor puede verificarlo fácilmente, pero implica más trabajo para el criptoanalista. Un método aún mejor es utilizar un hash criptográfico, concepto que exploraremos más adelante. Por el momento piense en ello como un CRC mejorado.

Si regresamos a la criptografía cuántica por un momento, también podemos ver que la redundancia juega un papel ahí. Debido a que Trudy interceptó los fotones, algunos bits en el relleno de una sola vez de Bob serán incorrectos. Bob necesita algo de redundancia en los mensajes entrantes para determinar que hay errores presentes. Una forma muy burda de redundancia es repetir dos veces el mensaje. Si ambas copias no son idénticas, Bob sabe que o la fibra tiene mucho ruido o que alguien está interviniendo la transmisión. Por supuesto que enviar todo el mensaje dos veces es excesivo; los códigos de Hamming o de Reed-Solomon constituyen formas más eficientes para realizar la detección y corrección de errores.

Pero debe quedar claro que para distinguir un mensaje válido de uno inválido, en especial en el caso de que haya un intruso activo, es necesaria cierta cantidad de redundancia.

Actualización

El segundo principio criptográfico es que se deben tomar medidas para asegurar que cada mensaje recibido se verifique con el fin de saber si está actualizado; es decir, que se haya enviado muy recientemente. Esta medida es necesaria para evitar que intrusos activos reproduzcan mensajes antiguos. Si no se tomaran tales medidas, nuestra ex empleada podría intervenir la línea telefónica de EP y simplemente continuar repitiendo los mensajes válidos enviados con anterioridad. En otras palabras, tenemos:

Principio criptográfico 2: Es necesario algún método para frustrar los ataques de repetición.

Una de tales medidas es incluir en cada mensaje una estampa de tiempo válida durante, digamos, 10 segundos. El receptor puede entonces guardar los mensajes unos 10 segundos y comparar los mensajes recién llegados con los anteriores para filtrar los duplicados. Los mensajes con una antigüedad mayor a 10 segundos pueden descartarse, dado que todas las repeticiones enviadas más de 10 segundos después también se rechazarán por ser demasiado viejas. Más adelante estudiaremos algunas medidas diferentes a las estampas de tiempo.

8.2 ALGORITMOS DE CLAVE SIMÉTRICA

La criptografía moderna usa las mismas ideas básicas que la criptografía tradicional (la transposición y la sustitución), pero su énfasis es distinto. Por tradición, los criptógrafos han usado algoritmos simples. Hoy día se hace lo opuesto: el objetivo es hacer el algoritmo de encriptación tan complicado y rebuscado que incluso si el criptoanalista obtiene cantidades enormes de texto cifrado a su gusto, no será capaz de entender nada sin la clave.

La primera clase de algoritmos de encriptación que analizaremos en este capítulo se conocen como **algoritmos de clave simétrica** porque utilizan la misma clave para encriptar y desencriptar. La figura 8-2 ilustra el uso de un algoritmo de clave simétrica. En particular, nos enfocaremos en los **sistemas de cifrado en bloques**, que toman un bloque de *n* bits de texto plano como entrada y lo transforman mediante el uso de la clave en un bloque de *n* bits de texto cifrado.

Los algoritmos criptográficos se pueden implementar ya sea en hardware (para velocidad) o en software (para flexibilidad). Aunque la mayoría de nuestro tratamiento tiene que ver con algoritmos y protocolos, que son independientes de la implementación real, no están de más unas cuantas palabras acerca de la construcción del hardware criptográfico. Las transposiciones y las sustituciones se pueden implementar mediante circuitos eléctricos sencillos. En la figura 8-6(a) se muestra un dispositivo, conocido como **caja P** (la P significa permutación), que se utiliza para efectuar una transposición de una entrada de 8 bits. Si los 8 bits se designan de arriba hacia abajo como 01234567, la salida de esta caja P en particular es 36071245. Mediante el cableado interno adecuado podemos hacer que una caja P efectúe cualquier transposición prácticamente a la velocidad de la luz, pues no se requiere ningún cálculo, sólo la propagación de la señal. Este diseño sigue el principio de Kerckhoff: el atacante sabe que el método general está permutando los bits. Lo que no sabe es qué bit va en qué lugar.

Las sustituciones se llevan a cabo mediante **cajas S**, como se muestra en la figura 8-6(b). En este ejemplo, se ingresa un texto plano de 3 bits y sale un texto cifrado de 3 bits. La entrada de 3 bits selecciona una de las ocho líneas de salida de la primera etapa y la establece en 1; todas las otras líneas son 0. La segunda etapa es una caja P. La tercera etapa codifica en binario nuevamente la línea de entrada

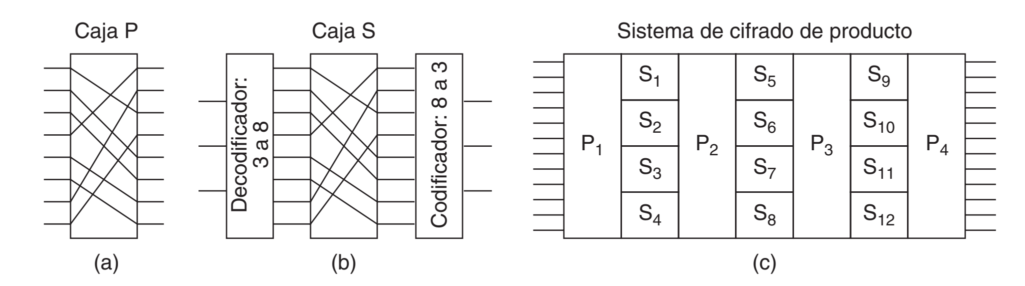


Figura 8-6. Elementos básicos de sistema de cifrado de producto. (a) Caja P. (b) Caja S. (c) Producto.

seleccionada. Con el cableado que se muestra, si los ocho números octales 01234567 entraran uno tras otro, la secuencia de salida sería de 24506713. En otras palabras, se ha reemplazado el 0 por el 2, el 1 por el 4, etc. De nuevo, podemos lograr cualquier sustitución mediante el cableado adecuado de la caja P dentro de la caja S. Además, es posible construir dicho dispositivo en hardware para alcanzar una gran velocidad, ya que los codificadores y decodificadores sólo tienen uno o dos retardos de compuerta lógica (subnanosegundos) y el tiempo de propagación a través de la caja P podría ser menor a 1 picosegundo.

El verdadero poder de estos elementos básicos sólo se hace aparente cuando ponemos en cascada una serie completa de cajas para formar un **sistema de cifrado de producto**, como se muestra en la figura 8-6(c). En este ejemplo, se trasponen (es decir, se permutan) 12 líneas de entrada en la primera etapa (P_1). En la segunda etapa, la entrada se divide en cuatro grupos de 3 bits, cada uno de los cuales se sustituye de manera independiente a los otros (S_1 a S_4). Este arreglo muestra un método para aproximar una caja S más grande a partir de varias cajas S más pequeñas. Es útil debido a que las cajas S pequeñas son prácticas para una implementación de hardware (por ejemplo, una caja S de 8 bits se puede realizar como una tabla de búsqueda de 256 entradas), pero las cajas S grandes son difíciles de construir (por ejemplo, una caja S de 12 bits necesitaría como mínimo $2^{12} = 4\,096$ alambres cruzados en su etapa media). Aunque este método es menos general, aún es poderoso. Mediante la inclusión de una cantidad suficiente de etapas en el sistema de cifrado de producto, la salida se puede convertir en una función excesivamente complicada de la entrada.

Los sistemas de cifrado de producto que operan en entradas de *k* bits para generar salidas de *k* bits son muy comunes. Por lo general, *k* puede valer de 64 a 256. Una implementación de hardware por lo general tiene cuando menos 10 etapas físicas, en lugar de sólo 7 como en la figura 8-6(c). Una implementación de software se programa como un ciclo con al menos 8 iteraciones, cada una de las cuales realiza sustituciones de tipo caja S en subloques del bloque de datos de 64 a 256 bits, seguido por una permutación que mezcla las salidas de las cajas S. Con frecuencia hay una permutación inicial especial y también una al final. En la literatura, las iteraciones se conocen como **rondas**.

8.2.1 DES: Estándar de Encriptación de Datos

En enero de 1977, el gobierno de Estado Unidos adoptó un sistema de cifrado de producto desarrollado por IBM como su estándar oficial para información no secreta. Este sistema de cifrado, **DES** (**Estándar de encriptación de datos**, del inglés *Data Encryption Standard*), se adoptó ampliamente en la industria para usarse en productos de seguridad. Ya no es seguro en su forma original, pero aún es útil en una forma modificada. Ahora explicaremos el funcionamiento del DES.

En la figura 8-7(a) se muestra un esbozo del DES. El texto plano se encripta en bloques de 64 bits, para producir 64 bits de texto cifrado. El algoritmo, que se parametriza mediante una clave de 56 bits, tiene 19 etapas diferentes. La primera etapa es una transposición, independiente de la clave, del texto

plano de 64 bits. La última etapa es el inverso exacto de esta transposición. La etapa previa a la última intercambia los 32 bits más a la izquierda con los 32 bits más a la derecha. Las 16 etapas restantes son funcionalmente idénticas, pero se parametrizan mediante diferentes funciones de la clave. El algoritmo se ha diseñado para permitir que la desencriptación se haga con la misma clave que la encriptación, una propiedad necesaria en cualquier algoritmo de clave simétrica. Los pasos simplemente se ejecutan en el orden inverso.

La operación de una de estas etapas intermedias se ilustra en la figura 8-7(b). Cada etapa toma dos entradas de 32 bits y produce dos salidas de 32 bits. La salida de la izquierda simplemente es una copia de la entrada de la derecha. La salida de la derecha es el XOR (OR exclusivo) a nivel de bits de la entrada izquierda y una función de la entrada derecha, además de ser la clave de esta etapa, K_i. Básicamente, toda la complejidad del algoritmo reside en esta función.

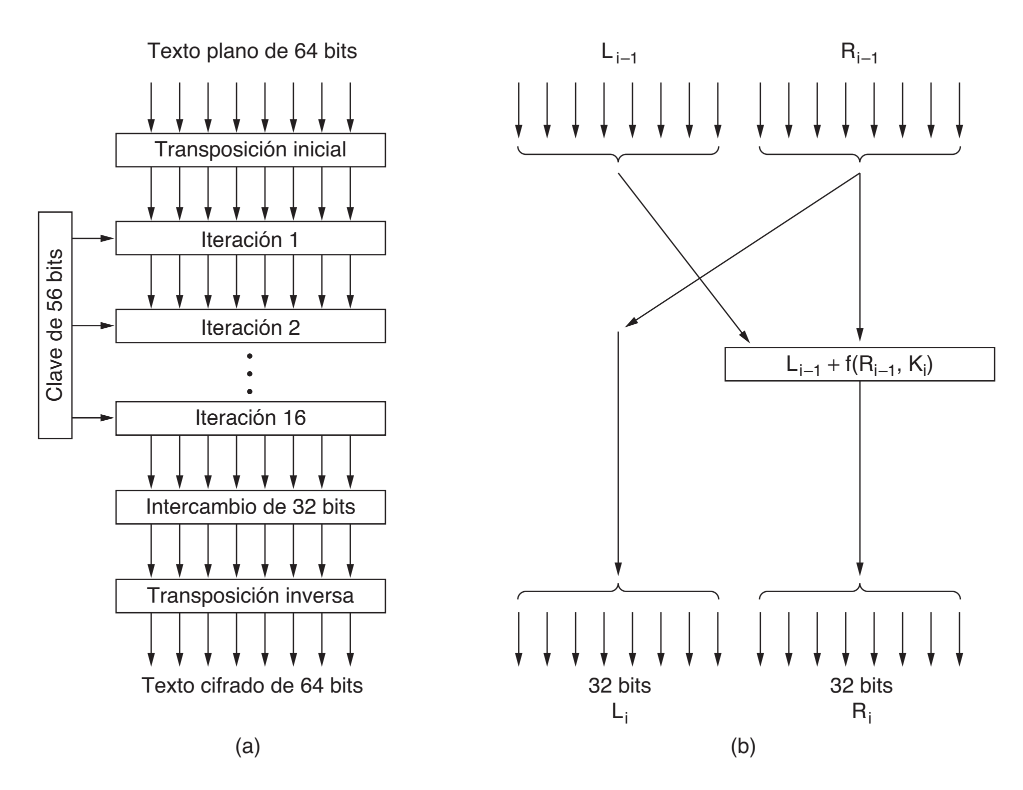


Figura 8-7. El Estándar de encriptación de datos. (a) Esbozo general. (b) Detalle de una iteración. El signo + encerrado en un círculo representa un OR exclusivo.

La función consiste en cuatro pasos, ejecutados en secuencia. Primero se construye un número de 48 bits, E, para lo cual se expande el R_{i-1} de 32 bits según una regla fija de transposición y duplicación. Segundo, se aplica un OR exclusivo a E y K_i. Esta salida entonces se divide en ocho grupos de 6 bits, cada uno de los cuales se alimenta a una caja S distinta. Cada una de las 64 entradas posibles a la caja S se asocia a una salida de 4 bits. Por último, estos 8×4 bits se pasan a través de una caja P.

En cada una de las 16 iteraciones, se usa una clave diferente. Antes de iniciarse el algoritmo, se aplica una transposición de 56 bits a la clave. Justo antes de cada iteración, la clave se divide en dos unidades de 28 bits, cada una de las cuales se rota hacia la izquierda una cantidad de bits dependiente del número de iteración. K_i se deriva de esta clave rotada al aplicarle otra transposición de 56 bits. En cada ronda un subgrupo de 48 bits diferente se extrae y permuta de los 56 bits.

Hay una técnica que algunas veces se utiliza para hacer a DES más fuerte, la cual se conoce como **blanqueamiento** (*whitening*). Consiste en aplicar un OR exclusivo a una clave aleatoria de 64 bits con cada bloque de texto plano antes de alimentarla al DES y después aplicar un OR exclusivo a una segunda clave de 64 bits con el texto cifrado resultante antes de transmitirla. El blanqueamiento se puede eliminar fácilmente mediante la ejecución de las operaciones inversas (si el receptor tiene las dos claves de blanqueamiento). Puesto que esta técnica agrega más bits a la longitud de la clave, provoca que una búsqueda completa del espacio de claves consuma mucho más tiempo. Observe que se utiliza la misma clave de blanqueamiento para cada bloque (es decir, sólo hay una clave de blanqueamiento).

El DES se ha visto envuelto en controversias desde el día en que se lanzó. Se basó en un sistema de cifrado desarrollado y patentado por IBM, llamado Lucifer, excepto que dicho sistema de cifrado usaba una clave de 128 bits en lugar de una de 56 bits. Cuando el gobierno federal de Estados Unidos quiso estandarizar un sistema de cifrado para uso no confidencial, "invitó" a IBM a "debatir" el asunto con la NSA, la dependencia de descifrado de códigos del gobierno estadounidense, que es el empleador de matemáticos y criptólogos más grande del mundo. La NSA es tan secreta que una broma de la industria versa así:

P: ¿Qué significa NSA?
R: Ninguna Supuesta Agencia.

En realidad, NSA significa Agencia Nacional de Seguridad, de Estados Unidos.

Tras estos debates, IBM redujo la clave de 128 a 56 bits y decidió mantener en secreto el proceso de diseño del DES. Mucha gente sospechó que la longitud de la clave se redujo para asegurar que la NSA pudiera descifrar el DES para que ninguna organización de menor presupuesto pudiera hacerlo. El objetivo del diseño secreto supuestamente era esconder una puerta trasera que pudiera facilitar aún más a la NSA el descifrado del DES. Cuando un empleado de la NSA pidió discretamente al IEEE que cancelara una conferencia planeada acerca de criptografía, eso incomodó aún más a la gente. La NSA negó todo.

En 1977, dos investigadores de criptografía de Stanford, Diffie y Hellman (1977) diseñaron una máquina para violar el DES y estimaron que el costo de la construcción podría ascender a unos 20 millones de dólares. Dado un trozo pequeño de texto plano y el texto cifrado correspondiente, esta máquina podría encontrar la clave mediante una búsqueda exhaustiva del espacio de claves de 2^{56} entradas en menos de un día. Hoy día el juego está en proceso. Tal máquina existe, está a la venta y su fabricación cuesta menos de $10000 (Kumar y colaboradores, 2006).

Triple DES

Ya en 1979, IBM se dio cuenta de que la longitud de la clave DES era muy corta y diseñó una forma de incrementarla de manera efectiva, mediante el uso de la triple encriptación (Tuchman, 1979). El método elegido, que desde entonces se ha incorporado en el Estándar Internacional 8732, se ilustra en la figura 8-8. Aquí se utilizan dos claves y tres etapas. En la primera etapa, el texto plano se encripta mediante DES de la forma usual con K_1. En la segunda etapa, el DES se ejecuta en modo de desencriptación, mediante el uso de K_2 como la clave. Por último, se realiza otra encriptación DES con K_1.

Este diseño da lugar inmediatamente a dos preguntas. Primera, ¿por qué sólo se usan dos claves en lugar de tres? Segunda, ¿por qué se utiliza **EDE** (**Encriptar Desencriptar Encriptar**) en lugar de **EEE** (**Encriptar Encriptar Encriptar**)? La razón de que se usen dos claves es que incluso los criptógrafos más paranoicos coinciden en que por ahora 112 bits son suficientes para las aplicaciones comerciales (y entre los criptógrafos la paranoia se considera una característica, no un error). Subir a 168 bits simplemente agregaría la sobrecarga innecesaria de administrar y transportar otra clave por muy poca ganancia.

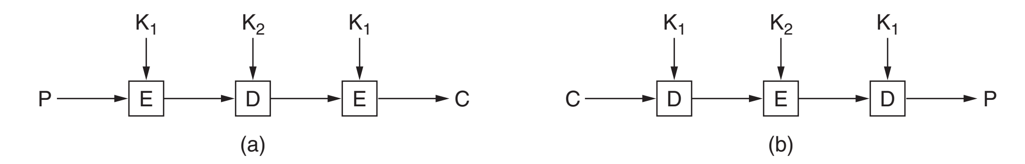


Figura 8-8. (a) Triple encriptación mediante el uso de DES. (b) Desencriptación.

La razón para encriptar, desencriptar y luego volver a encriptar es la compatibilidad hacia atrás con los sistemas DES de una sola clave. Tanto las funciones de encriptación como de desencriptación son asociaciones entre grupos de números de 64 bits. Desde el punto de vista criptográfico, las dos asociaciones son igualmente robustas. Sin embargo, mediante el uso de EDE en lugar de EEE, una computadora que emplea triple encriptación puede comunicarse con otra que usa encriptación sencilla con sólo establecer $K_1 = K_2$. Esta propiedad permite la introducción gradual de la triple encriptación, algo que no interesa a los criptógrafos académicos, pero que es de considerable importancia para IBM y sus clientes.

8.2.2 AES: Estándar de encriptación avanzada

Conforme el DES comenzó a acercarse al final de su vida útil, aun con el triple DES, el **NIST** (**Instituto Nacional de Estándares y Tecnología**, del inglés *Nacional Institute of Standards and Technology*), la agencia del Departamento de Comercio de Estados Unidos encargada de aprobar estándares del gobierno federal de ese país, decidió que el gobierno necesitaba un nuevo estándar criptográfico para uso no confidencial. El NIST estaba completamente consciente de toda la controversia alrededor del DES y sabía que si sólo anunciaba un nuevo estándar, todos los que tuvieran conocimiento sobre criptografía supondrían de manera automática que la NSA había construido una puerta trasera con el fin de leer todo lo que se encriptara con ella. Bajo estas condiciones, quizá nadie utilizaría el estándar y con seguridad tendría una muerte silenciosa.

Por esto, el NIST adoptó una estrategia sorprendentemente diferente para una burocracia gubernamental: promovió un concurso criptográfico. En enero de 1997, los investigadores de todo el mundo fueron invitados a emitir propuestas para un nuevo estándar, que se llamaría **AES** (**Estándar de encriptación Avanzada**, del inglés *Advanced Encryption Standard*). Las reglas fueron:

1. El algoritmo debe ser un sistema de cifrado de bloques simétrico.
2. Todo el diseño debe ser público.
3. Se deben soportar las longitudes de claves de 128, 192 y 256 bits.
4. Deben ser posibles las implementaciones tanto de software como de hardware.
5. El algoritmo debe ser público o con licencia en términos no discriminatorios.

Se realizaron 15 propuestas serias y se organizaron conferencias para presentarlas, en las cuales se alentó activamente a los asistentes a que encontraran errores en todas ellas. En agosto de 1998, el NIST seleccionó cinco finalistas, principalmente con base en su seguridad, eficiencia, simplicidad, flexibilidad y requerimientos de memoria (importantes para los sistemas integrados). Se llevaron a cabo más conferencias y se tomaron más críticas.

En octubre de 2000, el NIST anunció que había seleccionado a Rijndael, de Joan Daemen y Vincent Rijmen. El nombre Rijndael, que se pronuncia “Raindol” (más o menos), se deriva de los apellidos de los autores: Rijmen + Daemen. En noviembre de 2001, Rijndael se volvió el estándar de AES del gobierno

de Estados Unidos, publicado como FIPS (Estándar Federal para el Procesamiento de Información, del inglés *Federal Information Processing Standard*) 197. Debido a la extraordinaria apertura de la competencia, las propiedades técnicas de Rijndael y al hecho de que el equipo ganador estuvo compuesto por dos jóvenes criptógrafos belgas (quienes no es probable que hayan construido una puerta trasera sólo para complacer a la NSA), Rijndael se ha convertido en el estándar criptográfico dominante en el mundo. Ahora la encriptación y desencriptación AES forma parte del conjunto de instrucciones de algunos microprocesadores (por ejemplo, Intel).

Rijndael soporta longitudes de clave y tamaños de bloque de 128 a 256 bits en pasos de 32 bits. Las longitudes de clave y de bloque se pueden elegir de manera independiente. Sin embargo, el AES especifica que el tamaño de bloque debe ser de 128 bits y la longitud de clave debe ser de 128, 192 o 256 bits. Es poco probable que alguien utilice claves de 192 bits, por lo que, de hecho, el AES tiene dos variantes: un bloque de 128 bits con clave de 128 bits y un bloque de 128 bits con clave de 256 bits.

En el tratamiento del algoritmo examinaremos sólo el caso 128/128, porque es probable que éste se vuelva la norma comercial. Una clave de 128 bits da un espacio de claves de $2^{128} \approx 3 \times 10^{38}$ claves. Incluso si la NSA se las arregla para construir una máquina con 1 000 millones de procesadores paralelos, cada uno de los cuales es capaz de evaluar una clave por picosegundo, a tal máquina le llevaría aproximadamente 10^{10} años buscar en el espacio de claves. Para ese entonces el Sol ya se habrá apagado, por lo que las personas existentes tendrán que leer los resultados a la luz de las velas.

Rijndael

Desde una perspectiva matemática, Rijndael se basa en la teoría de campos de Galois, la cual da algunas propiedades verificables de seguridad. Sin embargo, también se puede ver como código C, sin entrar en las matemáticas.

Al igual que el DES, Rijndael utiliza sustituciones y permutaciones, así como múltiples rondas. El número de rondas depende del tamaño de clave y del tamaño de bloque, y es de 10 para las claves de 128 bits con bloques de 128 bits y aumenta hasta 14 para la clave o el bloque más grande. Sin embargo, a diferencia del DES, todas las operaciones involucran bytes completos para permitir implementaciones eficientes tanto en hardware como en software. En la figura 8-9 se muestra un esquema del código. Cabe mencionar que este código es para fines ilustrativos. Las buenas implementaciones de código de seguridad deben seguir prácticas adicionales, como poner en cero la memoria delicada una vez que se haya usado. Para ejemplos, consulte a Ferguson y colaboradores (2010).

La función *rijndael* tiene tres parámetros: *plaintext*, un arreglo de 16 bytes que contiene los datos de entrada; *ciphertext*, un arreglo de 16 bytes a donde se regresará la salida cifrada y *key*, la clave de 16 bytes. Durante el cálculo, el estado actual de los datos se mantiene en un arreglo de bytes llamado *state*, cuyo tamaño es *NROWS* × *NCOLS*. Para bloques de 128 bits, este arreglo es de 4 × 4 bytes. Con 16 bytes se puede almacenar el bloque de datos completo de 128 bits.

El arreglo *state* se inicializa con el texto plano y se modifica en cada paso en el cálculo. En algunos pasos se realiza la sustitución byte por byte. En otros, los bytes se permutan dentro del arreglo. También se utilizan otras transformaciones. Al final, el contenido de *state* se regresa como el texto cifrado.

El código empieza por expandir la clave en 11 arreglos del mismo tamaño que el estado. Éstos se almacenan en *rk*, que es un arreglo de estructuras, cada una de las cuales contiene un arreglo de estado. Uno de los arreglos se utilizará al inicio del cálculo y los otros 10 se utilizarán durante las 10 rondas, uno por ronda. El cálculo de las claves de ronda a partir de la clave de encriptación es muy complicado como para tratarlo aquí. Basta decir que las claves de ronda se producen mediante una rotación repetida y la aplicación de un OR exclusivo a varios grupos de bits de clave. Para todos los detalles, consulte a Daemen y Rijmen (2002).

```
#define LENGTH 16                                    /* # de bytes en el bloque de datos o en la clave */
#define NROWS 4                                      /* número de filas en estado */
#define NCOLS 4                                      /* número de columnas en estado */
#define ROUNDS 10                                    /* número de iteraciones */
typedef unsigned char byte;                          /* entero sin signo de 8 bits */

rijndael(byte plaintext[LENGTH], byte ciphertext[LENGTH], byte key[LENGTH])
{
  int r;                                             /* índice de ciclo */
  byte state[NROWS][NCOLS];                          /* estado actual */
  struct {byte k[NROWS][NCOLS];} rk[ROUNDS + 1];     /* claves de ronda */
  expand_key(key, rk);                               /* construye las claves de ronda */
  copy_plaintext_to_state(state, plaintext);         /* inicia el estado actual */
  xor_roundkey_into_state(state, rk[0]);             /* aplica XOR a la clave dentro del estado */
  for (r = 1; r <= ROUNDS; r++) {
    substitute(state);                               /* aplica la caja S a cada byte */
    rotate_rows(state);                              /* rota la fila i por i bytes */
    if (r < ROUNDS) mix_columns(state);              /* función de mezcla */
    xor_roundkey_into_state(state, rk[r]);           /* aplica XOR a la clave dentro del estado */
  }
  copy_state_to_ciphertext(ciphertext, state);       /* devuelve el resultado */
}
```

Figura 8-9. Un esquema de Rijndael en C.

El siguiente paso es copiar el texto plano en el arreglo *state* con el fin de poder procesarlo durante las rondas. Se copia en orden de columna; los primeros 4 bytes van en la columna 0, los siguientes 4 bytes en la columna 1, y así en lo sucesivo. Las columnas y las filas se numeran a partir de 0, aunque las rondas se numeran a partir de 1. En la figura 8-10 se muestra la configuración inicial de los 12 arreglos de bytes de tamaño 4×4.

Hay un paso adicional antes de que comience el cálculo principal: se aplica un OR exclusivo a *rk*[0] dentro de *state*, byte por byte. En otras palabras, cada uno de los 16 bytes de *state* se reemplaza con el resultado del OR exclusivo entre sí mismo y el byte correspondiente de *rk*[0].

Ahora es tiempo de presentar la atracción principal. El ciclo ejecuta 10 iteraciones, una por ronda, y transforma a *state* en cada iteración. El contenido de cada ronda se produce en cuatro pasos. El paso 1 realiza una sustitución byte por byte en *state*. Cada byte se utiliza a la vez como un índice en una caja S para reemplazar su valor por el contenido de entrada de esa caja S. Este paso es un sistema de cifrado de sustitución monoalfabética directo. A diferencia del DES, que tiene varias cajas S, Rijndael sólo tiene una caja S.

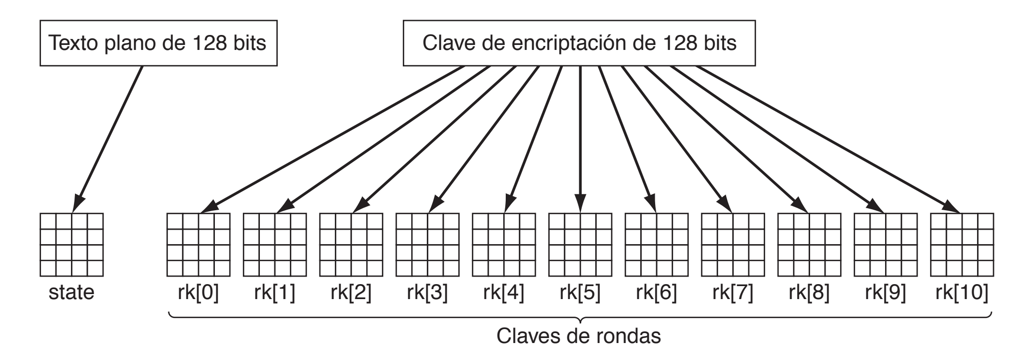


Figura 8-10. Creación de los arreglos *state* y *rk*.

El paso 2 rota a la izquierda cada una de las cuatro filas. La fila 0 se rota 0 bytes (es decir, no cambia), la fila 1 se rota 1 byte, la fila 2 se rota 2 bytes y la fila 3 se rota 3 bytes. Este paso disemina el contenido de los datos actuales alrededor del bloque, lo cual es análogo a las permutaciones de la figura 8-6.

El paso 3 mezcla cada una de las columnas independientemente de las demás. La mezcla se realiza mediante la multiplicación de matrices, en donde la nueva columna es el producto de la columna anterior y una matriz constante, y la multiplicación se realiza mediante el campo finito de Galois, $GF(2^8)$. Aunque esto podría sonar complicado, existe un algoritmo que permite calcular cada elemento de la nueva columna mediante dos tablas de búsquedas y tres OR exclusivos (Daemen y Rijmen, 2002, apéndice E).

Por último, el paso 4 aplica un OR exclusivo a la clave de esta ronda dentro del arreglo *state* para usarla en la siguiente.

Como cada paso es reversible, para realizar la desencriptación sólo hay que ejecutar el algoritmo en sentido inverso. Sin embargo, también hay un truco a través del cual se puede realizar la desencriptación mediante la ejecución del algoritmo de encriptación utilizando distintas tablas.

El algoritmo se ha diseñado no sólo para una gran seguridad, sino también para una gran velocidad. Una buena implementación de software en una máquina de 2 GHz debe tener la capacidad de alcanzar una tasa de encriptación de 700 Mbps, la cual es lo suficientemente rápida para encriptar cerca de 100 videos MPEG-2 en tiempo real. Las implementaciones de hardware son aún más rápidas.

8.2.3 Modos de sistema de cifrado

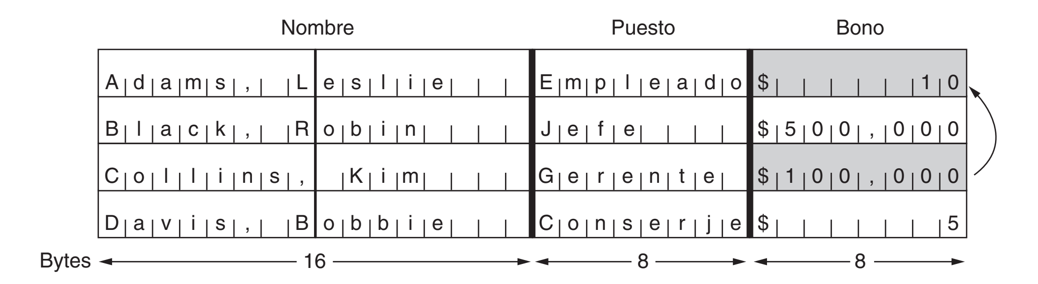
A pesar de toda esta complejidad, el AES (o el DES o, de hecho, cualquier sistema de cifrado de bloques) es básicamente un sistema de cifrado de sustitución monoalfabética que utiliza caracteres grandes (caracteres de 128 bits para el AES y caracteres de 64 bits para el DES). Siempre que el mismo bloque de texto plano entra por la parte frontal, el mismo bloque de texto cifrado sale por la parte posterior. Si encriptamos 100 veces el texto plano *abcdefgh* con la misma clave DES, obtendremos 100 veces el mismo texto cifrado. Un intruso puede explotar esta propiedad para poder destruir el sistema de cifrado.

Modo de Libro de Código Electrónico

Para ver cómo podemos utilizar esta propiedad de un sistema de cifrado de sustitución monoalfabética para quebrantar el sistema de cifrado en forma parcial, utilizaremos el DES (triple) porque es más fácil diseñar bloques de 64 bits que de 128 bits, pero el AES tiene exactamente el mismo problema. La forma directa de utilizar el DES para encriptar una pieza grande de texto plano es dividirla en bloques consecutivos de 8 bytes (64 bits) y luego encriptarlos uno tras otro con la misma clave. La última pieza de texto plano se rellena a 64 bits, en caso de ser necesario. Esta técnica se conoce como **modo ECB** (**modo de Libro de Código Electrónico**, del inlgés *Electronic Code Book mode*) en analogía con los libros de código anticuados, en los que se listaba cada palabra de texto plano, seguida por su texto cifrado (por lo general, un número decimal de cinco dígitos).

En la figura 8-11 se muestra el inicio de un archivo de computadora, en el cual se listan los bonos anuales que una compañía ha decidido otorgar a sus empleados. Este archivo consiste en registros consecutivos de 32 bytes, uno por empleado, en el formato que se muestra: 16 bytes para el nombre, 8 bytes para el puesto y 8 bytes para el bono. Cada uno de los 16 bloques de 8 bytes (numerados del 0 al 15) se encripta mediante DES (triple).

Leslie acaba de pelearse con el jefe y no espera un bono considerable. Por el contrario, Kim es la favorita del jefe y todos saben esto. Leslie puede obtener acceso al archivo después de encriptarlo, pero antes de enviarlo al banco. ¿Puede Leslie enmendar esta situación injusta, sólo con el archivo encriptado?



Nombre		Puesto	Bono
Adams, L	eslie	Empleado	$ 10
Black, R	obin	Jefe	$500,000
Collins,	Kim	Gerente	$100,000
Davis, B	obbie	Conserje	$ 5
Bytes 16		8	8

Figura 8-11. El texto plano de un archivo encriptado en forma de 16 bloques DES.

No hay problema. Todo lo que Leslie tiene que hacer es realizar una copia del decimosegundo bloque de texto cifrado (que contiene el bono de Kim) y utilizarlo para reemplazar el cuarto bloque de texto cifrado (que contiene el bono de Leslie). Incluso sin saber lo que dice el doceavo bloque, Leslie puede esperar una Navidad mucho más feliz este año (también existe la posibilidad de copiar el octavo bloque de texto cifrado, pero es más probable de detectar; además, Leslie no es una persona avara).

Modo de encadenamiento de bloques de sistema de cifrado

Para frustrar este tipo de ataque, podemos encadenar todos los sistemas de cifrado en bloques de varias formas; así, al reemplazar un bloque de la forma en que lo hizo Leslie, el texto plano desencriptado que se obtenga a partir del bloque reemplazado no servirá. Una forma de hacer esto es mediante el **encadenamiento de bloques de sistema de cifrado**. En este método, que se muestra en la figura 8-12, a cada bloque de texto plano se le aplica un OR exclusivo con el bloque anterior de texto cifrado antes de encriptarlo. En consecuencia, el mismo bloque de texto plano ya no se asocia al mismo bloque de texto cifrado, por lo que la encriptación deja de ser un enorme sistema de cifrado de sustitución monoalfabética. Al primer bloque se le aplica un OR exclusivo con un **IV** (**Vector de Inicialización**, del inglés *Initialization Vector*) elegido al azar, el cual se transmite (en texto plano) junto con el texto cifrado.

En el ejemplo de la figura 8-12 podemos ver la forma en que funciona el modo de encadenamiento de bloques de sistema de cifrado. Empezamos por calcular $C_0 = E(P_0 \text{ XOR } IV)$. Después calculamos $C_1 = E(P_1 \text{ XOR } C_0)$, y así sucesivamente. La desencriptación también utiliza un OR exclusivo para invertir el proceso, con $P_0 = IV \text{ XOR } D(C_0)$, etc. Observe que la encriptación del bloque i es una función de todo el texto plano de los bloques 0 a $i - 1$, por lo que el mismo texto plano genera un texto cifrado dependiendo de dónde ocurre. Una transformación del tipo que realizó Leslie no tendrá sentido para dos

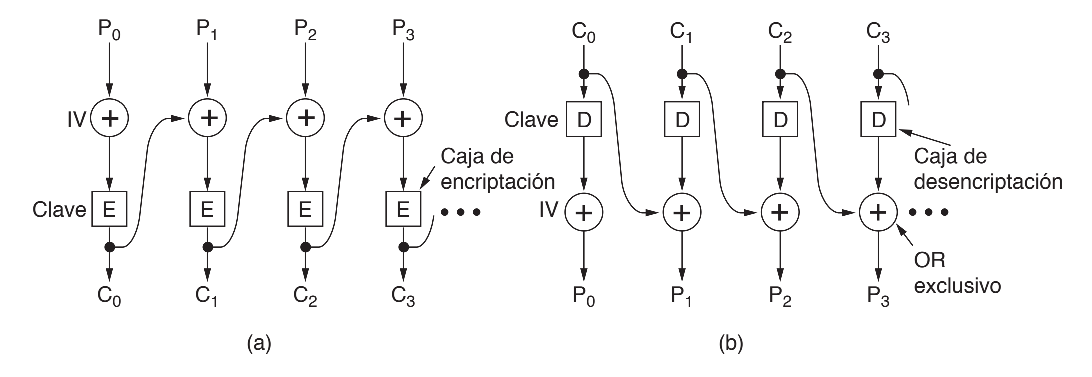


Figura 8-12. Encadenamiento de bloques de sistema de cifrado. (a) Encriptación. (b) Desencriptación.

bloques que inician en el campo de bono de Leslie. Para un oficial de seguridad astuto, esta peculiaridad podría sugerir en dónde comenzar la investigación resultante.

El encadenamiento de bloques de sistema de cifrado también tiene la ventaja de que el mismo bloque de texto plano no resultará en el mismo bloque de texto cifrado, lo que dificulta más el criptoanálisis. De hecho, ésta es la razón principal por la que se utiliza.

Modo de retroalimentación de sistema de cifrado

Sin embargo, el encadenamiento de bloques de sistema de cifrado tiene la desventaja de que se requiere la llegada de un bloque completo de 64 bits antes de que pueda comenzar la desencriptación. Para la encriptación byte por byte se utiliza el **modo de retroalimentación de sistema de cifrado** usando DES (triple), como se muestra en la figura 8-13. Para el AES la idea es exactamente la misma, sólo que se utiliza un registro de desplazamiento de 128 bits. En esta figura se muestra el estado de la máquina de encriptación después de haber encriptado y enviado los bytes 0 a 9. Cuando llega el byte 10 de texto plano, como se indica en la figura 8-13(a), el algoritmo DES opera sobre el registro de desplazamiento de 64 bits para generar un texto cifrado de 64 bits. El byte de más a la izquierda de ese texto cifrado se extrae y se le aplica un OR exclusivo con P_{10}. Ese byte se transmite por la línea de transmisión. Además, el registro de desplazamiento se desplaza 8 bits a la izquierda, lo cual provoca que C_2 salga del extremo izquierdo y que C_{10} se inserte en la posición que C_9 acaba de dejar libre en el extremo derecho.

Cabe mencionar que el contenido del registro de desplazamiento depende de la historia completa anterior del texto plano, por lo que un patrón que se repita varias veces en el texto plano se encriptará cada vez de manera distinta en el texto cifrado. Al igual que con el encadenamiento de bloques de sistemas de cifrado, se necesita un vector de inicialización para comenzar con el proceso.

La desencriptación con el modo de retroalimentación de un sistema cifrado funciona de la misma forma que la encriptación. En particular, el contenido del registro de desplazamiento se *encripta*, no se *desencripta*, por lo que el byte seleccionado al cual se le aplica el OR exclusivo con C_{10} para obtener P_{10} es el mismo al que se le aplicó el OR exclusivo con P_{10} para generar C_{10} en primer lugar. Mientras que los dos registros de desplazamiento sean idénticos, la desencriptación funcionará de manera correcta. Esto se ilustra en la figura 8-13(b).

Un problema con el modo de retroalimentación de sistema de cifrado es que si un bit del texto cifrado se invierte de manera accidental durante la transmisión, se dañarán los 8 bytes que se desencriptan mien-

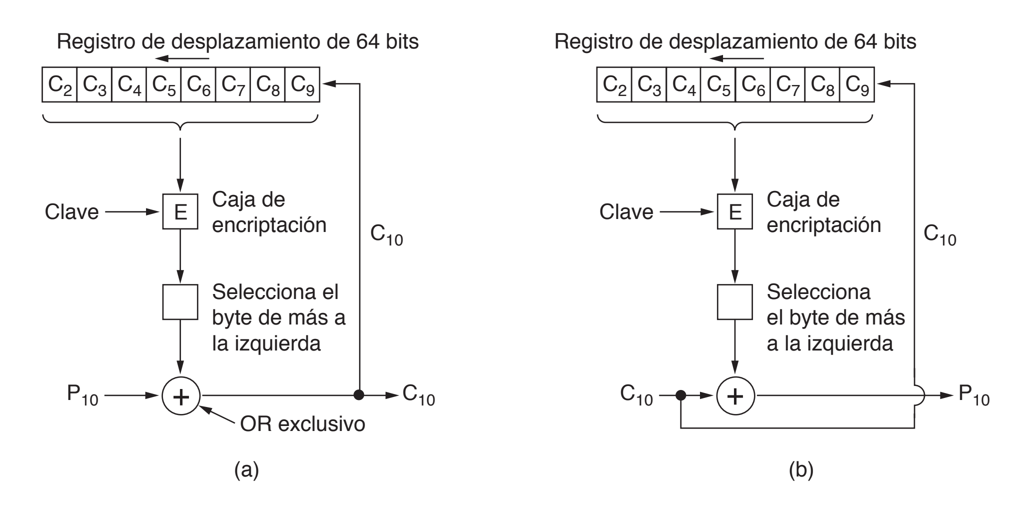


Figura 8-13. Modo de retroalimentación de sistema de cifrado. (a) Encriptación. (b) Desencriptación.

tras que el byte incorrecto se encuentre en el registro de desplazamiento. Una vez que el byte incorrecto se elimine del registro de desplazamiento, se generará nuevamente texto plano correcto. Por lo tanto, los efectos de un solo bit invertido se limitan relativamente a una ubicación específica y no se daña el resto del mensaje, pero sí se dañan los bits que contenga el registro de desplazamiento.

Modo de sistema de cifrado de flujo

Sin embargo, existen aplicaciones en las que tener un error de transmisión de 1 bit que arruina 64 bits de texto plano es un efecto demasiado grande. Para estas aplicaciones existe una cuarta opción: el **modo de sistema de cifrado de flujo**. Funciona encriptando un vector de inicialización, usando una clave para obtener un bloque de salida. Después, el bloque de salida se encripta usando la clave para obtener un segundo bloque de salida. Luego este bloque se encripta para obtener un tercer bloque, y así sucesivamente. La secuencia (arbitrariamente grande) de bloques de salida, llamada **flujo de claves**, se trata como un relleno de una sola vez y se le aplica un OR exclusivo con el texto plano para obtener el texto cifrado, como se muestra en la figura 8-14(a). Observe que el IV se utiliza sólo en el primer paso. Después de eso, se encripta la salida. También hay que tener en cuenta que el flujo de claves es independiente de los datos, por lo que se puede calcular por adelantado, en caso de ser necesario; además es completamente insensible a los errores de transmisión. En la figura 8-14(b) se muestra la desencriptación.

Para la desencriptación se genera el mismo flujo de claves en el lado receptor. Puesto que el flujo de claves depende sólo del IV y de la clave, no le afectan los errores de transmisión en el texto cifrado. Por lo tanto, un error de 1 bit en el texto cifrado transmitido genera sólo un error de 1 bit en el texto plano desencriptado.

Es esencial nunca utilizar dos veces el mismo par (clave, IV) con un sistema de cifrado de flujo, pues al hacerlo se generará cada vez el mismo flujo de claves. Si se utiliza el mismo flujo de claves dos veces, el texto cifrado queda expuesto a un **ataque de reutilización de flujo de claves**. Imagine que el bloque de texto plano, P_0, se encripta con el flujo de claves para obtener P_0 XOR K_0. Después, un segundo bloque de texto plano, Q_0, se encripta con el mismo flujo de claves para obtener Q_0 XOR K_0. Un intruso que capture estos dos bloques de texto cifrado sólo tiene que aplicarles un OR exclusivo en conjunto para obtener P_0 XOR Q_0, lo cual elimina la clave. Ahora el intruso tiene el OR exclusivo de los dos bloques de texto plano. Si se conoce uno de ellos o es posible adivinarlo, el otro también se puede descubrir. En cualquier caso, se puede atacar el OR exclusivo de dos flujos de texto plano mediante el uso de las propiedades estadísticas del mensaje. Por ejemplo, para texto en inglés, tal vez el carácter más común en el flujo será el OR exclusivo de dos espacios, seguido por el OR exclusivo de un espacio y la letra “e”, etc. En resumen, al poseer el OR exclusivo de dos bloques de texto plano, hay una excelente probabilidad de que el criptoanalista los deduzca.

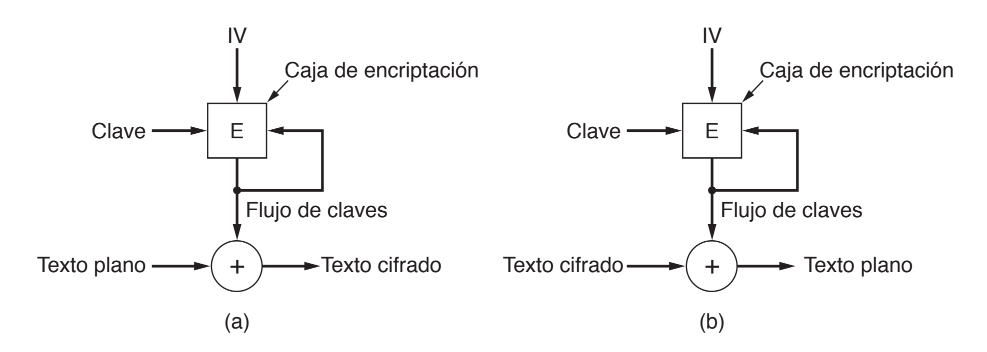


Figura 8-14. Un sistema de cifrado de flujo. (a) Encriptación. (b) Desencriptación.

Modo de contador

Un problema que todos los modos tienen, excepto el modo de libro de código electrónico, es que es imposible acceder de forma aleatoria a los datos encriptados. Por ejemplo, suponga que un archivo se transmite a través de una red y después se almacena en disco de forma encriptada. Ésta podría ser una forma razonable de operar si la computadora receptora fuera una computadora portátil con riesgo de ser robada. Si se almacenan todos los archivos importantes en forma encriptada, se reduce en gran medida el daño debido a la fuga de información confidencial en caso de que la computadora caiga en las manos equivocadas.

Sin embargo, es común acceder a los archivos de disco en un orden no secuencial, en especial los archivos en las bases de datos. En un archivo encriptado mediante el encadenamiento de bloques de sistema de cifrado, para acceder a un bloque aleatorio requiere primero desencriptar todos los bloques que estén delante de él, lo cual es muy complicado de lograr. Por esta razón, se ha inventado otro modo: **el modo de contador**, como se ilustra en la figura 8-15. En este caso, el texto plano no se encripta de manera directa, sino que se encripta el vector de inicialización más una constante, y al texto cifrado resultante se le aplica un OR exclusivo con el texto plano. Al incrementar en 1 el vector de inicialización para cada nuevo bloque, es más fácil desencriptar un bloque en cualquier parte del archivo sin antes tener que desencriptar todos sus predecesores.

Aunque el modo de contador es útil, tiene una debilidad que vale la pena mencionar. Suponga que en el futuro se utiliza de nuevo la misma clave, *K* (con un texto plano diferente pero con el mismo IV), y que un atacante adquiere todo el texto cifrado de ambas ejecuciones. Los flujos de claves son los mismos en ambos casos, por lo cual el sistema de cifrado se expone a un ataque de reutilización de flujo de claves del mismo tipo que vimos en los sistemas de cifrado de flujo. Todo lo que el criptoanalista tiene que hacer es aplicar un OR exclusivo a los dos textos cifrados en conjunto para eliminar toda la protección criptográfica y sólo obtener el OR exclusivo de los textos planos. Esta debilidad no significa que el modo de contador sea una mala idea. Simplemente significa que las claves y los vectores de inicialización se deben elegir de manera independiente y al azar. Incluso si se utiliza la misma clave dos veces de manera accidental, si el IV es diferente en cada ocasión, el texto plano estará seguro.

8.2.4 Otros sistemas de cifrado

DES y AES (Rijndael) son los algoritmos criptográficos de clave simétrica más conocidos, además de ser las opciones estándar en la industria, aunque sólo sea por cuestiones de responsabilidad (nadie lo culpará si usa AES en su producto y más adelante logran quebrantar este algoritmo, pero sin duda lo culparán si utiliza un sistema de cifrado no estándar y posteriormente lo quebrantan). Sin embargo, vale la pena men-

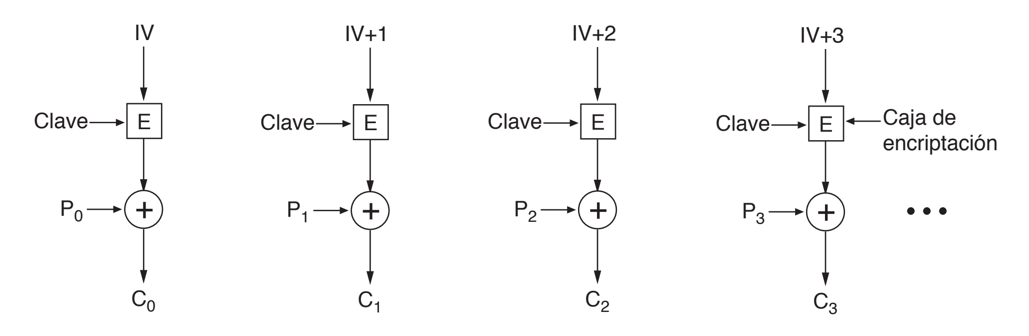


Figura 8-15. Encriptación mediante el uso del modo de contador.

cionar que se han diseñado muchos otros sistemas de cifrado de clave simétrica. Algunos de ellos están integrados en varios productos. En la figura 8-16 se listan algunos de los más comunes. Es posible utilizar combinaciones de estos sistemas de cifrado; por ejemplo, AES sobre Twofish, de manera que haya que quebrantar ambos sistemas de cifrado para poder recuperar los datos.

Sistema de cifrado	Autor	Longitud de clave	Comentarios
DES	IBM	56 bits	Muy débil para usarlo en la actualidad.
RC4	Ronald Rivest	1-2048 bits	Precaución: algunas claves son débiles.
RC5	Ronald Rivest	128-256 bits	Bueno, pero patentado.
AES (Rijndael)	Daemen y Rijmen	128-256 bits	La mejor opción.
Serpent	Anderson, Biham, Knudsen	128-256 bits	Muy sólido.
Triple DES	IBM	168 bits	Bueno, pero se está volviendo anticuado.
Twofish	Bruce Schneier	128-256 bits	Muy sólido; se utiliza mucho.

Figura 8-16. Algunos algoritmos criptográficos comunes de clave simétrica.

8.2.5 Criptoanálisis

Antes de dejar el tema de la criptografía de clave simétrica, vale la pena cuando menos mencionar cuatro avances del criptoanálisis. El primero es el **criptoanálisis diferencial** (Biham y Shamir, 1997). Podemos usar esta técnica para atacar cualquier sistema de cifrado en bloques. Primero empieza con un par de bloques de texto plano que difieren sólo en una cantidad pequeña de bits y observa con cuidado lo que ocurre en cada iteración interna a medida que avanza la encriptación. En muchos casos, algunos patrones son más comunes que otros, lo cual puede provocar ataques probabilísticos.

El segundo avance que vale la pena mencionar es el **criptoanálisis lineal** (Matsui, 1994). Éste puede descifrar el DES con sólo 2^{43} textos planos conocidos. Su función es aplicar un OR exclusivo a ciertos bits del texto plano y del texto cifrado en conjunto, para después examinar el resultado. Al hacerse en forma repetida, la mitad de los bits deben ser 0s y la otra mitad 1s. Sin embargo, con frecuencia los sistemas de cifrado introducen una desviación en una dirección o en la otra, y esta desviación, por pequeña que sea, se puede explotar para reducir el factor de trabajo. Para los detalles, consulte la publicación de Matsui.

El tercer avance es usar el análisis del consumo de energía eléctrica para averiguar las claves secretas. Las computadoras por lo general utilizan 3 volts para representar un bit 1 y 0 volts para representar un bit 0. Por lo tanto, para procesar un 1 se requiere más energía eléctrica que para procesar un 0. Si un algoritmo criptográfico consiste en un ciclo en el que los bits clave se procesan en orden, un atacante que reemplace el reloj principal de *n* GHz con uno lento (por ejemplo, 100 Hz) y coloque pinzas de caimán en las terminales de energía y tierra de la CPU, puede monitorear con precisión la energía consumida por cada instrucción de la máquina. A partir de estos datos, es sorprendente lo sencillo que es deducir la clave. Este tipo de criptoanálisis sólo se puede vencer si se codifica con cuidado el algoritmo en lenguaje ensamblador para asegurarse de que el consumo de energía sea independiente de la clave, así como de todas las claves de rondas individuales.

El cuarto avance es el análisis de temporización. Los algoritmos criptográficos están llenos de instrucciones if que prueban bits en las claves de ronda. Si las partes then y else tardan distintas cantidades de tiempo en ejecutarse, al reducir la velocidad del reloj y ver el periodo que tardan en ejecutarse varios pasos, también puede ser posible deducir las claves de ronda. Una vez que se conocen todas las claves de

ronda, por lo general es posible calcular la clave original. Los análisis de energía y temporización también se pueden utilizar de manera simultánea para facilitar el trabajo. Si bien los análisis de energía y temporización pueden parecer exóticos, en realidad son técnicas poderosas que pueden quebrantar cualquier sistema de cifrado que no esté diseñado de manera específica para resistirlas.

8.3 ALGORITMOS DE CLAVE PÚBLICA

A lo largo de la historia, el problema de distribución de claves siempre ha sido la parte más débil de la mayoría de los criptosistemas. Sin importar lo importante que fuera, si un intruso podía robar la clave, el sistema quedaba inutilizado. Los criptólogos siempre daban por hecho que las claves de encriptación y desencriptación eran iguales (o que se podían derivar fácilmente una de la otra). Pero la clave se tenía que distribuir a todos los usuarios del sistema. Por lo tanto, parecía haber un problema inherente. Era necesario proteger las claves contra robo, pero también se tenían que distribuir, por lo que no podían simplemente guardarse en una caja fuerte.

En 1976, dos investigadores de la Universidad de Stanford, Diffie y Hellman (1976), propusieron una clase totalmente nueva de criptosistema, en donde las claves de encriptación y desencriptación eran tan diferentes que no era posible derivar una a partir de la otra. En su propuesta, el algoritmo de encriptación (con clave) E y el algoritmo de desencriptación (con clave) D tenían que cumplir con tres requerimientos, los cuales se pueden expresar de la siguiente manera:

1. $D(E(P)) = P$.
2. Es demasiado difícil deducir D a partir de E.
3. E no se puede descifrar mediante un ataque de texto plano elegido.

El primer requerimiento dice que, si aplicamos D a un mensaje cifrado, $E(P)$, obtenemos de nuevo el mensaje de texto plano original, P. Sin esta propiedad, el receptor legítimo no podría desencriptar el texto cifrado. El segundo requerimiento no necesita explicación. El tercer requerimiento es necesario porque, como veremos en un momento, los intrusos pueden experimentar todo lo que deseen con el algoritmo. Bajo estas condiciones, no hay razón para que una clave de encriptación no se pueda hacer pública.

El método funciona de la siguiente manera. Una persona, por decir Alice, que quiera recibir mensajes secretos, primero diseña dos algoritmos que cumplan los requerimientos anteriores. Después el algoritmo de encriptación y la clave de Alice se hacen públicos, de ahí que se le llame **criptografía de clave pública**. Por ejemplo, Alice podría poner su clave pública en su página de inicio en la web. Utilizaremos la notación E_A para denotar el algoritmo de encriptación parametrizado por la clave pública de Alice. De manera similar, el algoritmo de desencriptación (secreto) parametrizado por la clave privada de Alice es D_A. Bob sigue el mismo procedimiento y hace a E_B pública, pero mantiene a D_B secreta.

Ahora veamos si podemos resolver el problema de establecer un canal seguro entre Alice y Bob, que nunca han tenido contacto previo. Se supone que tanto la clave de encriptación de Alice, E_A, como la clave de encriptación de Bob, E_B, se encuentran en archivos que todo el mundo puede leer. Ahora Alice toma su primer mensaje, P, calcula $E_B(P)$ y lo envía a Bob. A continuación, Bob lo desencripta mediante la aplicación de su clave secreta D_B [es decir, calcula $D_B(E_B(P)) = P$]. Nadie más puede leer el mensaje encriptado, $E_B(P)$, porque se supone que el sistema de encriptación es robusto y es demasiado difícil derivar D_B a partir de la clave E_B públicamente conocida. Para enviar una respuesta, R, Bob transmite $E_A(R)$. Ahora, Alice y Bob se pueden comunicar en forma segura.

Tal vez sea conveniente hacer una observación sobre la terminología. La criptografía de clave pública requiere que cada usuario tenga dos claves: una pública, que todo el mundo utiliza para encriptar los

mensajes a enviar a ese usuario, y una privada que el usuario necesita para desencriptar los mensajes. Nos referimos de manera consistente a estas claves como las claves *públicas* y *privadas*, respectivamente, y las distinguiremos de las claves *secretas* que se utilizan en la criptografía convencional de clave simétrica.

8.3.1 RSA

La única dificultad está en que necesitamos encontrar algoritmos que realmente satisfagan estos tres requerimientos. Debido a las ventajas potenciales de la criptografía de clave pública, muchos investigadores están trabajando de manera considerable en ello, y ya se han publicado algunos algoritmos. Un grupo del MIT (Rivest y colaboradores, 1978) descubrió un buen método, el cual se conoce por las iniciales de sus tres descubridores (Rivest, Shamir, Adleman): **RSA**. Este método ha sobrevivido a todos los intentos por quebrantarlo durante más de 30 años y se le considera muy sólido. Gran parte de la seguridad práctica se basa en él. Por esta razón, Rivest, Shamir y Adleman recibieron el Premio ACM Turing 2002. Su mayor desventaja es que requiere claves de por lo menos 1024 bits para una buena seguridad (en comparación con los 128 bits de los algoritmos de clave simétrica), por lo cual es muy lento.

El método RSA se basa en ciertos principios de la teoría de los números. Ahora resumiremos el uso del método; para los detalles, consulte la publicación original.

1. Seleccionar dos números primos grandes, p y q (por lo general de 1024 bits).
2. Calcular $n = p \times q$ y $z = (p - 1) \times (q - 1)$.
3. Seleccionar un número que sea relativamente primo para z y llamarlo d.
4. Encontrar e de tal forma que $e \times d = 1 \ mod \ z$.

Con estos parámetros calculados por adelantado, estamos listos para comenzar la encriptación. Dividimos el texto plano (que se considera como una cadena de bits) en bloques, para que cada mensaje de texto plano P se encuentre en el intervalo $0 \leq P < n$. Para hacer esto, es necesario agrupar el texto plano en bloques de k bits, donde k es el entero más grande para el cual $2^k < n$ es verdad.

Para encriptar un mensaje P, calculamos $C = P^e$ (mod n). Para desencriptar C, calculamos $P = C^d$ (mod n). Se puede demostrar que, para todos los P del intervalo especificado, las funciones de encriptación y desencriptación son inversas. Para ejecutar la encriptación, se necesitan e y n. Para llevar a cabo la desencriptación, se requieren d y n. Por lo tanto, la clave pública consiste en el par (e, n) y la clave privada consiste en (d, n).

La seguridad del método se basa en la dificultad para factorizar números grandes. Si el criptoanalista pudiera factorizar el valor de n (que se conoce públicamente), podría encontrar p y q y, a partir de éstos, z. Equipado con el conocimiento de z y de e, es posible encontrar d mediante el uso del algoritmo de Euclides. Por fortuna, los matemáticos han estado tratando de factorizar números grandes durante los últimos 300 años, y la evidencia acumulada sugiere que se trata de un problema muy difícil.

De acuerdo con Rivest y sus colegas, para factorizar un número de 500 dígitos se requerían 10^{25} años de tiempo de cómputo si se utilizara el algoritmo de fuerza bruta. En ambos casos se asumió el uso del mejor algoritmo conocido y una computadora con un tiempo de instrucción de 1 μseg. Con un millón de chips ejecutándose en paralelo, cada uno con un tiempo de instrucción de 1 nseg, de todas formas se requerirían 10^{16} años. Aun si las computadoras continúan aumentando su velocidad en un orden de magnitud cada década, pasarán muchos años antes de que sea posible factorizar un número de 500 dígitos, y para entonces nuestros descendientes simplemente podrán escoger valores de p y q todavía más grandes.

En la figura 8-17 se muestra un ejemplo pedagógico trivial de cómo funciona el algoritmo RSA. Para este ejemplo hemos seleccionado $p = 3$ y $q = 11$, lo cual nos da $n = 33$ y $z = 20$. Un valor ade-

Texto plano (P)			Texto cifrado (C)		Después de la desencriptación	
Simbólico	Numérico	P^3	P^3 (mod 33)	C^7	C^7 (mod 33)	Simbólico
S	19	6859	28	13492928512	19	S
U	21	9261	21	1801088541	21	U
Z	26	17576	20	1280000000	26	Z
A	01	1	1	1	01	A
N	14	2744	5	78125	14	N
N	14	2744	5	78125	14	N
E	05	125	26	8031810176	05	E
Cálculo del emisor				Cálculo del receptor		

Figura 8-17. Un ejemplo del algoritmo RSA.

cuado para d es $d = 7$, puesto que 7 y 20 no tienen factores comunes. Con base en estas opciones, podemos encontrar a e si resolvemos la ecuación $7e = 1$ (mod 20), que produce $e = 3$. El texto cifrado C, que corresponde a un mensaje de texto plano P, se obtiene mediante $C = P^3$ (mod 33). El receptor desencripta el texto cifrado mediante la regla $P = C^7$ (mod 33). En la figura se muestra como ejemplo la encriptación del texto plano "SUZANNE".

Dado que los números primos que se eligieron para este ejemplo son muy pequeños, P debe ser menor a 33, por lo que cada bloque de texto plano sólo puede contener un carácter. El resultado es un sistema de cifrado por sustitución monoalfabética, algo no muy impresionante. En cambio, si hubiéramos seleccionado p y $q \approx 2^{512}$, podríamos tener $n \approx 2^{1024}$, de tal manera que cada bloque podría ser de hasta 1024 bits o 128 caracteres de ocho bits, en comparación con ocho caracteres para el DES y 16 para el AES.

Cabe señalar que el uso del RSA como lo hemos descrito es semejante a usar un algoritmo simétrico en modo ECB: el mismo bloque de entrada produce el mismo bloque de salida. Por lo tanto, se requiere alguna forma de encadenamiento para la encriptación de datos. Sin embargo, en la práctica la mayoría de los sistemas basados en RSA usan criptografía de clave pública, principalmente para distribuir claves de sesión de una sola vez para usarlas con algún algoritmo de clave simétrica como el AES o el DES triple. El RSA es demasiado lento para poder encriptar grandes volúmenes de datos, pero se utiliza mucho para distribuir las claves.

8.3.2 Otros algoritmos de clave pública

Aunque el RSA se usa mucho, de ninguna manera es el único algoritmo de clave pública conocido. El primer algoritmo de clave pública fue el algoritmo de la mochila (Merkle y Hellman, 1978). La idea aquí es que alguien es propietario de una gran cantidad de objetos, cada uno con un peso distinto. Para codificar el mensaje, el dueño selecciona en secreto un subconjunto de los objetos y los coloca en la mochila. El peso total de los objetos contenidos en la mochila se hace público, así como la lista de todos los objetos posibles y sus correspondientes pesos. La lista de los objetos que se metieron en la mochila se mantiene en secreto. Con ciertas restricciones adicionales, el problema de determinar una lista posible de los objetos con el peso dado se consideró una operación computacional no viable, y formó la base del algoritmo de clave pública.

El inventor del algoritmo, Ralph Merkle, estaba bastante seguro de que no era posible quebrantar este algoritmo, por lo que ofreció una recompensa de 100 dólares a cualquiera que pudiera descifrarlo. En poco tiempo Adi Shamir (la "S" de RSA) lo descifró y cobró la recompensa. Sin desmotivarse, Merkle reforzó el algoritmo y ofreció una recompensa de 1000 dólares a quien pudiera quebrantar el nuevo algoritmo. Poco después Ronald Rivest (la "R" de RSA) lo descifró y cobró la recompensa.

Merkle no se atrevió a ofrecer 10 000 dólares para la siguiente versión, por lo que “A” (Leonard Adleman) no tuvo suerte. Sin embargo, el algoritmo de la mochila no se considera seguro, por lo que ya no se utiliza en la práctica.

Otros esquemas de clave pública se basan en la dificultad para calcular logaritmos discretos. Los algoritmos que usan este principio han sido inventados por El Gamal (1985) y Schnorr (1991).

Existen algunos otros esquemas, como los basados en curvas elípticas (Menezes y Vanstone, 1993), pero las dos categorías principales son las que se basan en la dificultad de factorizar números grandes y calcular logaritmos discretos mediante la operación módulo con un número primo grande. Estos problemas se consideran verdaderamente difíciles de resolver; los matemáticos han estado trabajando en ellos durante años sin obtener avances importantes.

8.4 FIRMAS DIGITALES

La autenticidad de muchos documentos legales, financieros y de otros tipos se determina mediante la presencia o ausencia de una firma manuscrita autorizada. Las fotocopias no cuentan. Para que los sistemas de mensajes computarizados reemplacen el transporte físico de papel y tinta, hay que encontrar un método que permita firmar documentos de una manera imposible de falsificar.

El problema de idear un reemplazo para las firmas manuscritas es difícil. En esencia, lo que se requiere es un sistema mediante el cual una parte pueda enviar un mensaje firmado a otra parte de modo que se cumplan las siguientes condiciones:

1. Que el receptor pueda verificar la identidad del transmisor.
2. Que el emisor no pueda repudiar más tarde el contenido del mensaje.
3. Que el receptor no haya podido elaborar el mensaje él mismo.

El primer requerimiento es necesario, por ejemplo, en los sistemas financieros. Cuando la computadora de un cliente ordena a la computadora de un banco que compre una tonelada de oro, ésta necesita asegurarse de que la computadora que da la orden realmente pertenece al cliente a cuya cuenta se le aplicará el cargo. En otras palabras, el banco tiene que autentificar la identidad del cliente (y éste a su vez tiene que autentificar al banco).

El segundo requerimiento es necesario para proteger al banco contra los fraudes. Suponga que el banco compra la tonelada de oro y justo después, el precio del oro cae drásticamente. Un cliente deshonesto podría entonces proceder a demandar al banco, alegando que nunca emitió una orden para comprar el oro. Cuando el banco presente el mensaje en la corte, el cliente puede negar haberlo enviado. La propiedad de que ninguna parte de un contrato pueda más tarde negar haber firmado se conoce como **no repudio**. Los esquemas de firma digital que estudiaremos a continuación ayudan a proporcionar dicha propiedad.

El tercer requerimiento es necesario para proteger al cliente en el caso de que el precio del oro se dispare y que el banco trate de construir un mensaje firmado en el que el cliente haya solicitado un lingote de oro en lugar de una tonelada. En este escenario fraudulento, el banco simplemente se queda con el resto del oro.

8.4.1 Firmas de clave simétrica

Una metodología para las firmas digitales es tener una autoridad central que sepa todo y en quien todos confíen; digamos el Gran Hermano (**Big Brother**, o *BB*). Así, cada usuario escoge una clave secreta y la

lleva personalmente a las oficinas del *BB*. Por ende, sólo Alice y el *BB* conocen la clave secreta de Alice, K_A, y así sucesivamente.

Cuando Alice desea enviar un mensaje de texto plano firmado (*P*) a su banquero Bob, genera $K_A(B, R_A, t, P)$, en donde *B* es la identidad de Bob, R_A es un número aleatorio elegido por Alice, *t* es una estampa de tiempo para asegurar que el mensaje sea reciente, y $K_A(B, R_A, t, P)$ es el mensaje encriptado con su clave, K_A. A continuación lo envía como se muestra en la figura 8-18. El *BB* ve que el mensaje proviene de Alice, lo desencripta y envía un mensaje a Bob como se muestra. El mensaje para Bob contiene el texto plano del mensaje de Alice y también el mensaje firmado $K_{BB}(A, t, P)$. Ahora, Bob se encarga de la solicitud de Alice.

¿Qué ocurre si más tarde Alice niega haber enviado el mensaje? El paso 1 es que todos demandan a todos (al menos en Estados Unidos). Por último, cuando el caso llega a la corte y Alice niega rotundamente haber enviado a Bob el mensaje en disputa, el juez pregunta a éste por qué está tan seguro de que el mensaje en disputa vino de Alice y no de Trudy. Bob primero indica que el *BB* no aceptaría un mensaje de Alice a menos que estuviera encriptado con K_A, por lo que no hay posibilidad de que Trudy enviara al *BB* un mensaje falso de Alice sin que el *BB* lo detectara de inmediato.

A continuación Bob presenta de manera dramática la prueba A: K_{BB} (A, t, P); y dice que éste es un mensaje firmado por el *BB* para comprobar que Alice envió *P* a Bob. El juez entonces pide al *BB* (en quien todo el mundo confía) que desencripte la prueba A. Cuando el *BB* testifica que Bob dice la verdad, el juez se pronuncia a favor de Bob. Caso cerrado.

Un problema potencial con el protocolo de firma de la figura 8-18 es que Trudy reproduzca cualquiera de los dos mensajes. Para minimizar este problema, en todos los intercambios se usan estampas de tiempo. Además, Bob puede revisar todos los mensajes recientes para ver si se utilizó R_A en cualquiera de ellos. De ser así, el mensaje se descarta como repetición. Observe que con base en la estampa de tiempo, Bob rechazará los mensajes muy viejos. Para protegerse contra ataques de repetición instantánea, simplemente examina el R_A de cada mensaje entrante para ver si se ha recibido dicho mensaje de Alice durante la hora pasada. Si no, Bob puede suponer con seguridad que ésta es una solicitud nueva.

8.4.2 Firmas de clave pública

Un problema estructural con el uso de la criptografía de clave simétrica para las firmas digitales es que todos tienen que estar de acuerdo en confiar en el Gran Hermano. Es más, el Gran Hermano puede leer todos los mensajes firmados. Los candidatos más lógicos para operar el servidor del Gran Hermano son: el gobierno, los bancos, los contadores y los abogados. Por desgracia, ninguna de estas organizaciones inspira una total confianza a todos los ciudadanos. Por tanto, sería bueno si la firma de documentos no requiriera una autoridad de confianza.

Por fortuna, la criptografía de clave pública puede hacer una contribución importante en esta área. Vamos a suponer que los algoritmos de encriptación y desencriptación de clave pública tienen la propiedad de que $E(D(P)) = P$, además (desde luego) de la propiedad normal de que $D(E(P)) = P$. (RSA tiene

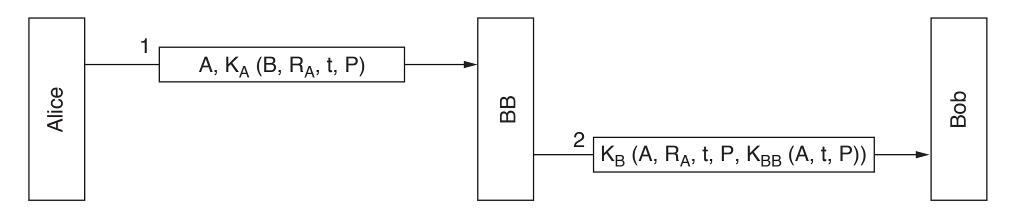


Figura 8-18. Firmas digitales con el Gran Hermano (*Big Brother*).

esta propiedad, por lo que el supuesto es razonable.) Suponiendo que éste sea el caso, Alice puede enviar un mensaje de texto plano firmado (P) a Bob al transmitir $E_B(D_A(P))$. Observe bien que Alice conoce su propia clave (privada), D_A, así como la clave pública de Bob, E_B, por lo que Alice puede elaborar este mensaje.

Cuando Bob recibe el mensaje, lo transforma mediante su clave privada de la manera usual, para producir $D_A(P)$ como se muestra en la figura 8-19. Bob almacena este texto en un lugar seguro y después aplica E_A para obtener el texto plano original.

Para ver cómo funciona la propiedad de firma, suponga que más tarde Alice niega haber enviado el mensaje P a Bob. Cuando el caso llega a la corte, Bob puede presentar tanto P como $D_A(P)$. El juez puede comprobar fácilmente que Bob tiene un mensaje válido encriptado por D_A con sólo aplicarle E_A. Puesto que Bob no conoce la clave privada de Alice, la única forma en que Bob pudo haber adquirido un mensaje encriptado por esa clave sería que Alice en efecto lo hubiera enviado. Mientras esté en la cárcel por perjurio y fraude, Alice tendrá tiempo suficiente para diseñar algoritmos de clave pública nuevos e interesantes.

Aunque el uso de la criptografía de clave pública para las firmas digitales es un esquema elegante, hay problemas relacionados con el entorno en el que operan más que con el algoritmo básico. Por una parte, Bob puede demostrar que Alice envió un mensaje siempre y cuando D_A permanezca en secreto. Si Alice divulga su clave secreta, el argumento ya no es válido, puesto que cualquiera pudo haber enviado el mensaje, incluido el mismo Bob.

El problema podría surgir, por ejemplo, si Bob es el corredor de Bolsa de Alice. Suponga que ella le indica a Bob que compre ciertas acciones o bonos. Inmediatamente después, el precio cae de manera drástica. Para repudiar el mensaje que envió a Bob, corre a la policía y les informa que robaron su casa junto con la PC que contiene su clave. Dependiendo de las leyes de su estado o país, ella puede o no ser responsable legalmente, en especial si indica no haber descubierto el robo sino hasta que llegó de trabajar, varias horas después del supuesto incidente.

Otro problema con el esquema de firmas es lo que ocurre si Alice decide cambiar su clave. Sin duda es legal hacerlo, y tal vez sea conveniente cambiar la clave en forma periódica. Si luego surge un caso en la corte, como se describió antes, el juez aplicará la E_A actual a $D_A(P)$ y descubrirá que no produce P. Bob quedará en ridículo en ese momento.

En principio es posible utilizar cualquier algoritmo de clave pública para firmas digitales. El estándar de facto de la industria es el algoritmo RSA. Muchos productos de seguridad lo usan. Sin embargo, en 1991, el NIST (Instituto Nacional de Estándares y Tecnología) propuso el uso de una variación del algoritmo de clave pública de El Gamal para su nuevo **DSS** (**Estándar de Firmas Digitales**, del inglés *Digital Signatura Standard*). La seguridad de El Gamal radica en la dificultad para calcular logaritmos discretos, en vez de la dificultad para factorizar números grandes.

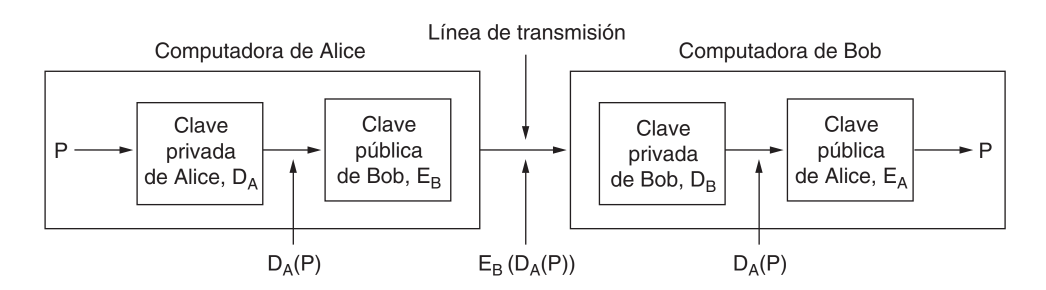


Figura 8-19. Firmas digitales mediante el uso de la criptografía de clave pública.

Como es normal cuando el gobierno intenta dictar estándares criptográficos, hubo una protesta general. El DSS recibió críticas por ser

1. Demasiado secreto (el NSA diseñó el protocolo para usar El Gamal).
2. Demasiado lento (de 10 a 40 veces más lento que RSA para comprobar firmas).
3. Demasiado nuevo (El Gamal no se ha analizado de manera exhaustiva).
4. Demasiado inseguro (clave fija de 512 bits).

En una revisión posterior, el cuarto punto se prestó a discusión cuando se permitieron claves de hasta 1024 bits. Sin embargo, los primeros dos puntos permanecen válidos.

8.4.3 Resúmenes de mensaje

Una crítica a los métodos de firma es que con frecuencia combinan dos funciones distintas: autentificación y confidencialidad. A menudo se requiere autentificación, pero no confidencialidad. Asimismo, con frecuencia es más fácil obtener una licencia de exportación si el sistema en cuestión sólo proporciona autentificación pero no confidencialidad. A continuación describiremos un esquema de autentificación que no requiere la encriptación del mensaje completo.

Este esquema se basa en la idea de una función de *hash* unidireccional que toma una parte arbitrariamente grande de texto plano y, a partir de ella, calcula una cadena de bits de longitud fija. Esta función de hash, *MD*, que se conoce comúnmente como **resumen de mensaje** (*message digest*), tiene cuatro propiedades importantes:

1. Dado P, es fácil calcular $MD(P)$.
2. Dado $MD(P)$, es en efecto imposible encontrar P.
3. Dado P, nadie puede encontrar P' de tal manera que $MD(P') = MD(P)$.
4. Un cambio en la entrada de incluso 1 bit produce una salida muy diferente.

Para cumplir con el criterio 3, la función de hash debe tener al menos 128 bits de longitud, y de preferencia más. Para cumplir con el criterio 4, la función de hash debe truncar los bits en forma minuciosa, no como los algoritmos de encriptación de clave simétrica que hemos visto.

Es mucho más rápido calcular el compendio de mensaje a partir de una pieza de texto plano que encriptar ese texto plano con un algoritmo de clave pública, por lo que es posible utilizar los resúmenes de mensaje para acelerar los algoritmos de firma digital. Para ver cómo funciona esto, considere una vez más el protocolo de firma de la figura 8-18. En lugar de firmar P con K_{BB} (A, t, P), el *BB* ahora calcula el resumen de mensaje mediante la aplicación de *MD* a P para producir $MD(P)$. A continuación, el *BB* incluye $K_{BB}(A, t, MD(P))$ como quinto elemento de la lista encriptada con K_B que se envía a Bob, en vez de K_{BB} (A, t, P).

Si surge una disputa, Bob puede presentar tanto P como K_{BB} $(A, t, MD(P))$. Una vez que el Gran Hermano (*Big Brother*) desencripta el mensaje para el juez, Bob tiene $MD(P)$, el cual se garantiza que es genuino, junto con el supuesto P. Dado que es prácticamente imposible que Bob encuentre otro mensaje que dé este resultado de la función de hash, el juez se convencerá fácilmente de que dice la verdad. Al usar los resúmenes de mensaje de esta manera, obtenemos un ahorro tanto en el tiempo de encriptación como en los costos de transporte de los mensajes.

Los resúmenes de mensaje funcionan también en los criptosistemas de clave pública, como se muestra en la figura 8-20. Aquí, Alice primero calcula el resumen de mensaje de su texto plano. Después firma el resumen de mensaje y lo envía a Bob junto con el el texto plano. Si Trudy reemplaza P en el camino, Bob verá esto cuando él mismo calcule $MD(P)$.

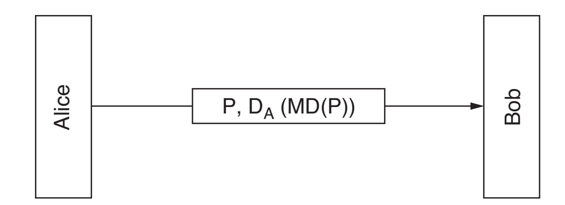


Figura 8-20. Firmas digitales mediante el uso de resúmenes de mensaje.

SHA-1 y SHA-2

Se han propuesto una variedad de funciones para el resumen de mensajes. Una de las más utilizada es **SHA-1** (**Algoritmo de Hash seguro 1**, del inglés *Secure Hash Algorithm 1*) (NIST, 1993). Al igual que todos los resúmenes de mensaje, opera truncando los bits de una manera lo bastante complicada como para que cada bit de salida se vea afectado por cada bit de entrada. La NSA desarrolló el algoritmo SHA-1; el NIST lo bendijo en el FIPS 180-1. Procesa datos de entrada en bloques de 512 bits y genera un resumen de mensaje de 160 bits. En la figura 8-21 se ilustra una forma típica para que Alice envíe a Bob un mensaje no secreto, pero firmado. Aquí su mensaje de texto plano se alimenta en el algoritmo SHA-1 para obtener un hash SHA-1 de 160 bits. A continuación Alice firma el hash con su clave privada RSA y lo envía a Bob junto con el mensaje de texto plano.

Después de recibir el mensaje, Bob calcula el hash SHA-1 por su cuenta y también aplica la clave pública de Alice al hash firmado para obtener el hash original, *H*. Si los dos concuerdan, el mensaje se considera válido. Puesto que no hay forma de que Trudy modifique el mensaje (texto plano) mientras está en tránsito y produzca uno nuevo que haga hash a *H*, Bob puede detectar con facilidad cualquier cambio que Trudy haya hecho al mensaje. Para los mensajes cuya integridad es importante pero cuyo contenido no es secreto, el esquema de la figura 8-21 se utiliza mucho. Por un costo de cómputo relativamente bajo, garantiza que cualquier modificación hecha al mensaje de texto plano en tránsito pueda detectarse con una probabilidad muy alta.

Ahora veamos de manera breve cómo funciona SHA-1. Empieza por rellenar el mensaje al agregar 1 bit al final, seguido de tantos bits 0 como sean necesarios (pero como mínimo 64) para que la longitud sea un múltiplo de 512 bits. A continuación, al número de 64 bits que contiene la longitud del mensaje antes del relleno se le aplica un OR en los 64 bits de menor orden. En la figura 8-22, el mensaje se muestra con un relleno a la derecha debido a que el texto y las figuras van de izquierda a derecha (es decir, por lo general el extremo inferior derecho se percibe como el final de la figura). En las computadoras esta orientación corresponde a máquinas *big-endian* como la SPARC, la IBM 360 y sus sucesores, pero SHA-1 siempre rellena el final del mensaje, sin importar cuál máquina endian se utilice.

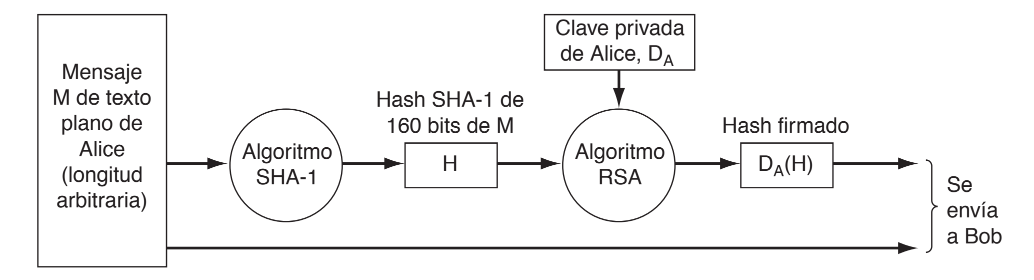


Figura 8-21. Uso de SHA-1 y RSA para firmar mensajes no secretos.

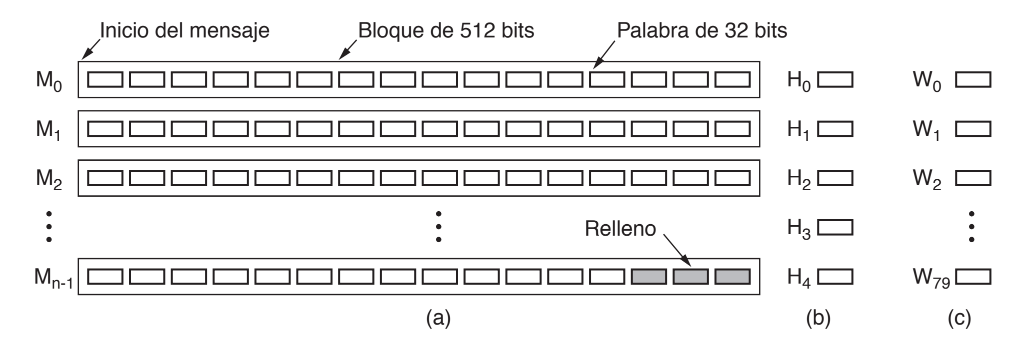


Figura 8-22. (a) Relleno de un mensaje a un múltiplo de 512 bits. (b) Las variables de salida. (c) El arreglo de palabras.

Durante el cálculo, SHA-1 mantiene cinco variables de 32 bits, H_0 a H_4, donde se acumula el hash. Dichas variables se muestran en la figura 8-22(b). Se inicializan con constantes especificadas en el estándar.

Ahora se procesa cada uno de los bloques M_0 a M_{n-1}. Para el bloque actual, las 16 palabras primero se copian al inicio de un arreglo auxiliar de 80 palabras, W, como se muestra en la figura 8-22(c). Después las otras 64 palabras de W se rellenan mediante la fórmula

$$W_i = S^1 (W_{i-3} \text{ XOR } W_{i-8} \text{ XOR } W_{i-14} \text{ XOR } W_{i-16}) \qquad (16 \leq i \leq 79)$$

en donde $S^b(W)$ representa la rotación circular izquierda de la palabra de 32 bits, W, por b bits. Ahora cinco variables de trabajo A a E se inicializan desde H_0 hasta H_4, respectivamente.

Podemos expresar el cálculo real en pseudo-C de la siguiente forma:

```
for (i = 0; i < 80; i++) {
   temp=S5(A) + fi(B,C,D) + E + Wi + Ki;
   E = D; D = C; C = S30(B); B = A; A = temp;
}
```

en donde las constantes K_i se definen en el estándar. Las funciones de mezcla f_i se definen como

$$f_i (B, C, D) = (B \text{ AND } C) \text{ OR } (\text{NOT } B \text{ AND } D) \qquad (0 \leq i \leq 19)$$
$$f_i (B, C, D) = B \text{ XOR } C \text{ XOR } D \qquad (20 \leq i \leq 39)$$
$$f_i (B, C, D) = (B \text{ AND } C) \text{ OR } (B \text{ AND } D) \text{ OR } (C \text{ AND } D) \qquad (40 \leq i \leq 59)$$
$$f_i (B, C, D) = B \text{ XOR } C \text{ XOR } D \qquad (60 \leq i \leq 79)$$

Cuando se completan las 80 iteraciones del ciclo, las variables A a E se agregan a las variables H_0 a H_4, respectivamente.

Ahora que se ha procesado el primer bloque de 512 bits, se inicia el siguiente. El arreglo W se reinicia desde el nuevo bloque, pero H se deja como estaba. Al terminar este bloque se inicia el siguiente, y así en lo sucesivo hasta que todos los bloques de mensajes de 512 bits se hayan procesado por completo. Al terminar el último bloque, las cinco palabras de 32 bits en el arreglo H se envían a la salida como el hash criptográfico de 160 bits. El código de C completo para SHA-1 se proporciona en el RFC 3174.

Se han desarrollado nuevas versiones de SHA-1 que producen hashes de 224, 256, 384 y 512 bits. En conjunto, a estas versiones se les conoce como SHA-2. No sólo son estos hashes más largos que los hashes del SHA-1, sino que también se modificó la función de resumen para combatir ciertas debilidades potenciales del SHA-1. El SHA-2 aún no se utiliza mucho, pero es probable que sea muy utilizado en lo futuro.

MD5

Para completar el tema, mencionaremos otro resumen popular. El **MD5** (Rivest, 1992) es el quinto de una serie de resúmenes de mensaje diseñados por Ronald Rivest. En esencia, el mensaje se rellena a una longitud de 448 bits (módulo 512). Después la longitud original del mensaje se adjunta como un entero de 64 bits para proporcionar una entrada total cuya longitud sea múltiplo de 512 bits. Cada ronda del cálculo requiere un bloque de 512 bits de entrada y lo mezcla a conciencia con un búfer de 128 bits en ejecución. Para una buena medida, la mezcla usa una tabla que se construye a partir de la función seno. El objetivo de usar una función conocida es para evitar cualquier sospecha de que el diseñador haya construido una puerta trasera por la que sólo él pueda entrar. Este proceso continúa hasta que se hayan consumido todos los bloques de entrada. El contenido del búfer de 128 bits forma el resumen de mensaje.

Después de más de una década de uso y estudio continuo, las debilidades en el MD5 han originado la habilidad de encontrar colisiones, o distintos mensajes con el mismo hash (Sotirov y colaboradores, 2008). Ésta es la sentencia de muerte para una función de resumen, ya que significa que el resumen no se puede usar en forma segura para representar un mensaje. Por lo tanto, la comunidad de seguridad considera que el MD5 se ha quebrado; se debe reemplazar siempre que sea posible y ningún sistema nuevo debe usarlo como parte de su diseño. Sin embargo, el MD5 aún se utiliza en algunos sistemas existentes.

8.4.4 El ataque de cumpleaños

En el mundo de la criptografía, nada es lo que parece. Podríamos pensar que se requieren alrededor de 2^m operaciones para destruir un resumen de mensaje de m bits. De hecho, con frecuencia $2^{m/2}$ operaciones son suficientes si se usa el **ataque de cumpleaños**, un enfoque publicado por Yuval (1979) en su ahora clásico trabajo *How to Swindle Rabin* (Cómo estafar a Rabin).

La idea de este ataque proviene de una técnica que los profesores de matemáticas usan con frecuencia en sus cursos de probabilidad. La pregunta es: ¿cuántos estudiantes se necesitan en un grupo antes de que la probabilidad de tener dos personas con el mismo cumpleaños sea mayor a 1/2? Muchos estudiantes suponen que la respuesta debe ser mucho mayor que 100. De hecho, la teoría de la probabilidad indica que es de apenas 23. Sin hacer un análisis riguroso, intuitivamente, con 23 personas, podemos formar $(23 \times 22)/2 = 253$ pares diferentes, cada uno de los cuales tiene una probabilidad de 1/365 de cumplir el requisito. Visto así, la respuesta ya no es en realidad tan sorprendente.

En términos más generales, si hay alguna correspondencia entre las entradas y las salidas con n entradas (gente, mensajes, etc.) y k salidas posibles (cumpleaños, resúmenes de mensaje, etc.), hay $n(n-1)/2$ pares de entradas. Si $n(n-1)/2 > k$, la posibilidad de que cuando menos una coincida es bastante buena. Por lo tanto, en términos aproximados es probable una coincidencia para $n > \sqrt{k}$. Este resultado significa que tal vez sea posible quebrantar un resumen de mensaje de 64 bits si se generan aproximadamente 2^{32} mensajes y se buscan dos con el mismo resumen de mensaje.

Veamos ahora un ejemplo práctico. El Departamento de Ciencias Computacionales de la Universidad Estatal tiene un puesto para un profesor vitalicio y dos candidatos: Tom y Dick. Tom fue contratado dos años antes que Dick, por lo que su candidatura será considerada antes. Si Tom obtiene el puesto, será mala suerte para Dick. Tom sabe que la jefa del departamento, Marilyn, tiene un buen concepto de su trabajo, por lo que le pide que escriba una carta de recomendación para el rector, quien decidirá sobre el caso de Tom. Una vez enviadas, todas las cartas se vuelven confidenciales.

Marilyn le dice a su secretaria, Ellen, que escriba una carta al rector, delineando lo que quiere en ella. Cuando está lista, Marilyn la revisará, calculará y firmará el resumen de 64 bits y la enviará al rector. Ellen puede mandar la carta después por correo electrónico.

Por desgracia para Tom, Ellen tiene una relación sentimental con Dick y le gustaría dejar fuera a Tom, por lo que escribe la siguiente carta con las 32 opciones entre corchetes:

Estimado rector Smith,

Esta [*carta* | *mensaje*] es para dar mi opinión [*honesta* | *franca*] sobre el profesor Tom Wilson, que [*ahora* | *este año*] [*es candidato a* | *está en espera de*] un puesto docente. He [*conocido a* | *trabajado con*] el profesor Wilson durante [*cerca de* | *casi*] seis años. Es un investigador [*sobresaliente* | *excelente*] de gran [*talento* | *habilidad*], [*mundialmente* | *internacionalmente*] conocido por sus [*brillantes* | *creativas*] investigaciones sobre [*muchos* | *una gran variedad de*] problemas [*difíciles* | *desafiantes*].

Él es también un [*profesor* | *educador*] [*altamente* | *muy*] [*respetado* | *admirado*]. Sus estudiantes han hecho evaluaciones [*sobresalientes* | *espectaculares*] de sus [*clases* | *cursos*]. Es el [*profesor* | *instructor*] [*más admirado* | *más querido*] [*del departamento* | *por nosotros*].

[*Además* | *Por otra parte*], el profesor Wilson es [*hábil* | *diligente*] para recaudar fondos. Sus [*subvenciones* | *contratos*] han aportado una cantidad [*importante* | *sustancial*] de dinero [*al* | *a nuestro*] Departamento. [*Este dinero ha* | *Estos fondos han*] [*posibilitado* | *permitido*] que [*emprendamos* | *pongamos en práctica*] muchos programas [*especiales* | *importantes*], [*como* | *por ejemplo*] su programa Estado 2000. Sin estos fondos [*seríamos incapaces de* | *no podríamos*] continuar este programa, que es tan [*importante* | *esencial*] para ambos. Recomiendo encarecidamente que se le otorgue el puesto docente.

Por desgracia para Tom, tan pronto como Ellen termina de redactar y mecanografiar esta carta, también escribe una segunda:

Estimado rector Smith,

Esta [*carta* | *mensaje*] es para dar mi opinión [*honesta* | *franca*] sobre el profesor Tom Wilson, que [*ahora* | *este año*] [*es candidato a* | *está en espera de*] un puesto docente. He [*conocido a* | *trabajado con*] Tom durante [*cerca de* | *casi*] seis años. Es un investigador [*malo* | *mediocre*] poco conocido en su [*campo* | *área*]. Sus investigaciones [*casi nunca* | *pocas veces*] muestran [*entendimiento* | *comprensión de*] los problemas [*clave* | *principales*] [*del día* | *de nuestros tiempos*].

Además, no es un [*profesor* | *educador*] [*respetado* | *admirado*]. Sus estudiantes han hecho evaluaciones [*pobres* | *malas*] de sus [*clases* | *cursos*]. Es el [*profesor* | *instructor*] menos querido [*del Departamento* | *por nosotros*], conocido [*principalmente* | *más*] en [*el* | *nuestro*] Departamento por su [*tendencia* | *propensión*] a [*ridiculizar* | *avergonzar*] a los estudiantes lo bastante [*tontos* | *imprudentes*] como para hacer preguntas durante su clase.

[*Además* | *Por otra parte*], Tom es [*malo* | *marginal*] para recaudar fondos. Sus [*subvenciones* | *contratos*] han aportado una cantidad [*precaria* | *insignificante*] de dinero [*al* | *a nuestro*] Departamento. A menos que [*se recolecte dinero* | *se consigan fondos*] pronto, tendremos que cancelar algunos programas esenciales, como su programa Estado 2000. Por desgracia, en estas [*condiciones* | *circunstancias*] no puedo recomendarlo de buena [*conciencia* | *fe*] a usted para [*el puesto docente* | *un puesto permanente*].

Ahora Ellen prepara su computadora para calcular los 2^{32} resúmenes de mensaje para cada carta durante la noche. Con suerte, un resumen de la primera carta será igual a un resumen de la segunda. De no serlo puede agregar algunas opciones más e intentar de nuevo esta noche. Suponga que encuentra una correspondencia. Llamemos a la carta "buena" *A* y a la "mala" *B*.

A continuación, Ellen envía la carta *A* a Marilyn para su aprobación. Mantiene la carta *B* en secreto, sin mostrársela a nadie. Marilyn, por supuesto, la aprueba, calcula su resumen de mensaje de 64 bits, lo firma y lo manda por correo electrónico al rector Smith. Luego Ellen envía por separado la carta *B* al rector (no la carta *A*, que se suponía debía haber enviado).

Al recibir la carta y el resumen de mensaje firmado, el rector ejecuta el algoritmo de resumen de mensaje sobre la carta *B*, constata que coincide con lo que Marilyn le envió y despide a Tom. El rector no se da cuenta de que Ellen se las ingenió para generar dos cartas con el mismo resumen de mensaje y le envió una diferente de la que Marilyn vio y aprobó. (Desenlace opcional: Ellen le dice a Dick lo que hizo.

Dick se horroriza y rompe con ella. Ellen se pone furiosa y confiesa su falta a Marilyn. Ésta llama al rector. Tom consigue el puesto docente a fin de cuentas). Con el SHA-1, el ataque de cumpleaños es difícil porque incluso a la tremenda velocidad de un billón de resúmenes por segundo, se requerirían más de 32 000 años para calcular los 2^{80} resúmenes de dos cartas con 80 variantes cada una, e incluso entonces no se garantiza una coincidencia. Claro que con una nube de 1 000 000 de chips trabajando en paralelo, 32 000 años se convierten en dos semanas.

8.5 ADMINISTRACIÓN DE CLAVES PÚBLICAS

Mediante la criptografía de clave pública, las personas que no comparten una clave común de todos modos se pueden comunicar en forma segura. También es posible firmar mensajes sin la presencia de un tercero de confianza. Por último, gracias a los resúmenes de mensajes firmados el receptor puede verificar de una manera fácil y segura la integridad de los mensajes recibidos.

Sin embargo, hay un problema que hemos pasado por alto demasiado rápido: si Alice y Bob no se conocen entre sí, ¿cómo obtiene cada uno la clave pública del otro para iniciar el proceso de comunicación? La solución más obvia —colocar su clave pública en su sitio web— no funciona por la siguiente razón. Suponga que Alice quiere buscar la clave pública de Bob en el sitio web de él. ¿Cómo lo hace? Comienza tecleando el URL de Bob. A continuación su navegador busca la dirección DNS de la página de inicio de Bob y le envía una solicitud GET, como se muestra en la figura 8-23. Por desgracia, Trudy intercepta la solicitud y responde con una página de inicio falsa, quizás una copia de la de Bob excepto porque reemplaza la clave pública de Bob con la de Trudy. Cuando Alice encripta su primer mensaje mediante E_T, Trudy lo desencripta, lo lee, lo vuelve a encriptar con la clave pública de Bob y lo envía a éste, quien no tiene la menor idea de que Trudy está leyendo los mensajes que le llegan. Peor aún, Trudy puede modificar los mensajes antes de volverlos a encriptar para Bob. Es evidente que se necesita un mecanismo para asegurar que las claves públicas se puedan intercambiar de manera segura.

8.5.1 Certificados

Como un primer intento por distribuir claves públicas de manera segura, podemos imaginar un centro de distribución de claves KDC disponible en línea las 24 horas del día, que proporciona claves públicas bajo demanda. Uno de los muchos problemas con esta solución es que no es escalable, y el centro de distribución de claves podría convertirse rápidamente en un “cuello de botella”. Además, si alguna vez fallara, la seguridad en Internet se paralizaría por completo.

Por estas razones se desarrolló una solución diferente, una que no requiere que el centro de distribución de claves esté en línea todo el tiempo. De hecho, ni siquiera tiene que estar en línea. En su lugar, lo que hace es certificar las claves públicas que pertenecen a las personas, empresas y otras organizaciones.

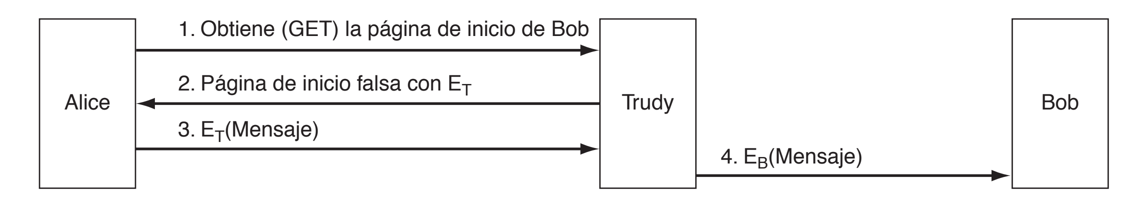


Figura 8-23. Una forma en la que Trudy puede destruir la encriptación de clave pública.

Ahora, a una organización que certifica claves públicas se le conoce como **CA** (**Autoridad de Certificación**, del inglés *Certification Authority*).

Como un ejemplo, suponga que Bob desea permitir que Alice y otras personas que no conoce se comuniquen con él de manera segura. Él puede ir con la CA con su clave pública, junto con su pasaporte o licencia de conducir para pedir su certificación. A continuación, la CA emite un certificado similar al que se muestra en la figura 8-24 y firma su hash SHA-1 con la clave privada de la CA. Luego Bob paga la cuota de la CA y obtiene un CD-ROM que contiene el certificado con su hash firmado.

```
Certifico que la clave pública
   19836A8B03030CF83737E3837837FC3s87092827262643FFA82710382828282A
pertenece a
   Robert John Smith
   12345 University Avenue
   Berkeley, CA 94702
   Cumpleaños: Julio 4, 1958
   Correo electrónico: bob@superdupernet.com
```

Hash SHA-1 del certificado anterior firmado con la clave privada de la CA

Figura 8-24. Un posible certificado y su hash firmado.

El trabajo fundamental de un certificado es enlazar una clave pública con el nombre de un personaje principal (individuo, empresa, etc.). Los certificados no son secretos ni están protegidos. Por ejemplo, Bob podría tomar la decisión colocar su nuevo certificado en su sitio web, con un vínculo en la página de inicio que diga "Haga clic aquí para obtener mi certificado de clave pública". El clic resultante podría regresar el certificado y el bloque de la firma (el hash SHA-1 firmado del certificado).

Ahora vayamos otra vez al escenario que se muestra en la figura 8-23. Cuando Trudy intercepta la solicitud que Alice realiza para obtener la página de inicio de Bob, ¿qué puede hacer Trudy? Puede poner su propio certificado y bloque de firma en la página falsificada, pero cuando Alice lea el certificado, verá inmediatamente que no está hablando con Bob porque el nombre de éste no se encuentra en dicho certificado. Trudy puede modificar sobre la marcha la página de inicio de Bob y reemplazar la clave privada de Bob con la suya. Sin embargo, cuando Alice ejecute el algoritmo SHA-1 sobre el certificado, obtendrá un hash que no corresponde con el que obtiene al aplicar la clave pública reconocida de la CA al bloque de firma. Puesto que Trudy no tiene la clave privada de la CA, no tiene forma de generar un bloque de firma que contenga el hash de la página web modificada con su clave pública en él. De esta manera, Alice puede estar segura de que tiene la clave pública de Bob y no la de Trudy o la de alguien más. Y, como prometimos, este esquema no requiere que la CA esté en línea para la verificación, por lo que se elimina un potencial "cuello de botella".

Mientras que la función estándar de un certificado es enlazar una clave pública a un personaje principal, un certificado también se puede utilizar para enlazar una clave pública a un **atributo**. Por ejemplo, un certificado podría decir: "Esta clave pública pertenece a alguien mayor de 18 años". Podría utilizarse para probar que el dueño de la clave privada no es una persona menor de edad y se le permita acceder a cierto material no apto para niños, entre otras cosas, pero sin revelar la identidad del dueño. Por lo general, la persona que tiene el certificado podría enviarlo al sitio web, al personaje principal o al proceso que se preocupa por la edad. El sitio, personaje principal o proceso podría generar a continuación un número aleatorio y encriptarlo con la clave pública del certificado. Si el dueño pudiera desencriptarlo y regresarlo, ésa sería una prueba de que el dueño tenía el atributo establecido en el certificado. De manera alternativa, el número aleatorio podría utilizarse para generar una clave de sesión para la conversación resultante.

Otro ejemplo en el que un certificado podría contener un atributo es un sistema distribuido orientado a objetos. Por lo general, cada objeto tiene varios métodos. El dueño del objeto podría proporcionar a cada cliente un certificado que dé un mapa de bits de cuáles métodos puede invocar y que enlace dicho mapa de bits a una clave pública mediante un certificado firmado. De nuevo, si el dueño del certificado puede probar la posesión de la clave privada correspondiente, se le permitirá ejecutar los métodos en el mapa de bits. Esta metodología tiene la propiedad de que no es necesario conocer la identidad del dueño, una propiedad útil en las situaciones en las que la privacidad es importante.

8.5.2 X.509

Si todas las personas que quisieran algo firmado fueran a la CA con un tipo diferente de certificado, la administración de todos los distintos formatos pronto se volvería un problema. Para resolverlo se ha diseñado un estándar para certificados, aprobado por la ITU. Dicho estándar se conoce como **X.509** y se utiliza mucho en Internet. Ha pasado por tres versiones desde su estandarización inicial en 1988. Aquí analizaremos la versión V3.

El X.509 ha recibido una enorme influencia del mundo de OSI, y ha tomado prestadas algunas de sus peores características (por ejemplo, la asignación de nombres y la codificación). Lo sorprendente es que la IETF estuvo de acuerdo con el X.509, aun cuando en casi todas las demás áreas, desde direcciones de máquinas y protocolos de transporte hasta los formatos de correo electrónico, la IETF por lo general ignoró a la OSI y trató de hacerlo bien. La versión del X.509 de la IETF se describe en el RFC 5280.

En esencia, el X.509 es una forma de describir certificados. Los campos principales en un certificado se listan en la figura 8-25. Las descripciones que se establecen en esa figura deben proporcionar una idea general de lo que hacen los campos. Para información adicional, por favor consulte el estándar mismo o el RFC 2459.

Por ejemplo, si Bob trabaja en el Departamento de Préstamos del Banco Monetario, su dirección X.500 podría ser:

```
/C=MX/O=BancoMonetario/OU=Prestamo/CN=Bob/
```

Campo	Significado
Versión	Qué versión de X.509.
Número de serie	Este número más el nombre de la CA identifican el certificado de manera única.
Algoritmo de firma	El algoritmo utilizado para firmar el certificado.
Emisor	El nombre X.500 de la CA.
Periodo de validez	Los tiempos inicial y final del periodo de validez.
Nombre del sujeto	La entidad cuya clave se va a certificar.
Clave pública	La clave pública del sujeto y la ID del algoritmo que la utiliza.
ID del emisor	Un ID opcional que identifica al emisor del certificado en forma única.
ID del sujeto	Un ID opcional que identifica al sujeto del certificado en forma única.
Extensiones	Se han definido muchas extensiones.
Firma	La firma del certificado (firmada por la clave privada de la CA).

Figura 8-25. Los campos básicos de un certificado X.509.

donde *C* corresponde al país, *O* a la organización, *OU* a la unidad organizacional y *CN* a un nombre común. Las CA y otras entidades se nombran de forma similar. Un problema considerable con los nombres X.500 es que, si Alice está tratando de contactar a *bob@bancomonetario.com* y se le asigna un certificado con un nombre X.500, tal vez no sea obvio para ella que el certificado se refiera al Bob que ella busca. Por fortuna, a partir de la versión 3 se permiten los nombres DNS en lugar de los nombres X.500, por lo que este problema se desvanecerá en un momento dado.

Los certificados están codificados mediante la **ASN.1** (**Notación de Sintaxis Abstracta 1**, del inglés *Abstract Syntax Notation 1*) de la OSI, que puede considerarse como si fuera una estructura de C, pero con una notación muy peculiar y prolija. Es posible encontrar información acerca de X.509 en Ford y Baum (2000).

8.5.3 Infraestructuras de clave pública

El hecho de que una sola CA emitiera todos los certificados del mundo obviamente no funcionaría. Podría derrumbarse por la carga y también podría ser un punto central de fallas. Una posible solución sería tener múltiples autoridades CA que fueran operadas por la misma organización y que utilizaran la misma clave privada para firmar los certificados. Si bien esto podría solucionar los problemas de carga y de fallas, introduciría un nuevo problema: la fuga de claves. Si hubiera docenas de servidores esparcidos por todo el mundo, todos con la misma clave privada de la CA, la probabilidad de que esta clave fuera robada o se filtrara de algún otro modo se incrementaría de manera considerable. Puesto que la situación comprometida de esta clave arruinaría la infraestructura de la seguridad electrónica mundial, tener una sola CA central es muy peligroso.

Además, ¿qué organización podría operar la CA? Es difícil imaginar cualquier autoridad que se pudiera aceptar mundialmente como legítima y digna de confianza. En algunos países las personas insistirían en que fuera el gobierno, mientras que en otros rechazarían esta opción por completo.

Por estas razones se ha desarrollado una forma diferente para certificar claves públicas. Su nombre general es **PKI** (**Infraestructura de Clave Pública**, del inglés *Public Key Infrastructure*). En esta sección resumiremos cómo funciona en general, aunque se han generado diversas propuestas, por lo que es probable que los detalles cambien con el tiempo.

Una PKI tiene varios componentes: usuarios, autoridades de certificación (CA), certificados y directorios. Lo que la PKI hace es proporcionar una forma de estructurar estos componentes y definir estándares para los diversos documentos y protocolos. Una forma particularmente simple de PKI es una jerarquía de autoridades CA, como se muestra en la figura 8-26. En este ejemplo mostramos tres niveles, pero en la práctica podrían ser menos o más. La CA de nivel superior, la raíz, certifica a las autoridades CA de segundo nivel, a las que llamaremos autoridades **RA** (**Autoridades Regionales**, del inglés *Regional Authorities*) debido a que podrían cubrir alguna región geográfica, como un país o un continente. Sin embargo, este término no es estándar; de hecho, ningún término es realmente estándar para los diversos niveles del árbol. A su vez, estas autoridades RA certifican a las CA reales, las cuales emiten los certificados X.509 a organizaciones e individuos. Cuando la raíz autoriza una nueva RA, genera un certificado X.509 donde indica que ha aprobado la RA, e incluye en él la nueva clave pública de la RA, la firma y se la entrega a la RA. De manera similar, cuando una RA aprueba una nueva CA, produce y firma un certificado que indica su aprobación y que contiene la clave pública de la CA.

Nuestra PKI funciona como se muestra a continuación. Suponga que Alice necesita la clave pública de Bob para comunicarse con él, por lo que busca y encuentra un certificado que la contenga, firmado por la CA 5. Sin embargo, Alice nunca ha escuchado acerca de la CA 5. En lo que a ella respecta, la CA 5 podría ser la hija de 10 años de Bob. Podría ir con la CA 5 y decirle: “Prueba tu autenticidad”. La CA 5 le responderá con el certificado que obtuvo de la RA 2, el cual contiene la clave pública de la CA 5.

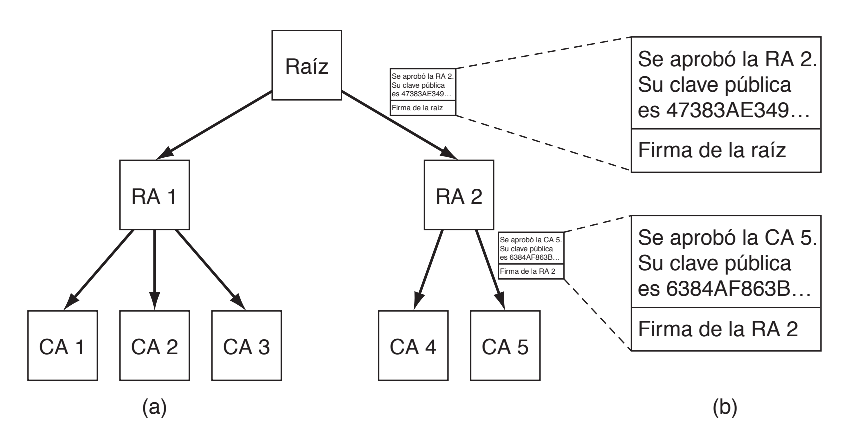


Figura 1-16. (a) Una PKI jerárquica. (b) Una cadena de certificados.

Una vez que tenga la clave pública de la CA 5, Alice podrá verificar que el certificado de Bob realmente fue firmado por la CA 5 y que, por lo tanto, es legal.

A menos que la RA 2 sea el hijo de 12 años de Bob. Así, el siguiente paso es que Alice pida a la RA 2 que pruebe su autenticidad. La respuesta a su consulta es un certificado firmado por la raíz que contiene la clave pública de la RA 2. Ahora Alice está segura de que tiene la clave pública de Bob.

Pero, ¿cómo averigua Alice la clave pública de la raíz? Magia. Se supone que todos conocen la clave pública de la raíz. Por ejemplo, su navegador podría tener integrada la clave pública de la raíz.

Bob es una persona amigable y no desea causar a Alice tanto trabajo. Él sabe que ella va a tener que verificar la CA 5 y la RA 2, por lo que para ahorrarle algunos problemas, recolecta los dos certificados necesarios y se los proporciona junto con el de él. Ahora ella puede utilizar el conocimiento que tiene de la clave pública de la raíz para verificar el certificado de nivel superior y la clave pública contenida ahí para verificar la segunda. De esta manera, Alice no necesita contactar a nadie para realizar la verificación. Puesto que todos los certificados están firmados, Alice puede detectar con facilidad cualquier intento de alterar el contenido. En ocasiones, a una cadena de certificados que va de esta forma a la raíz se denomina **cadena de confianza** o **ruta de certificación**. La técnica se utiliza ampliamente en la práctica.

Por supuesto, aún tenemos el problema de quién va a ejecutar la raíz. La solución no es tener una sola raíz, sino tener muchas, cada una con sus propias autoridades RA y CA. De hecho, los navegadores modernos vienen precargados con claves públicas para cerca de 100 raíces, algunas veces llamadas **anclas de confianza**. De esta forma, es posible evitar tener una sola autoridad mundial de confianza.

Pero ahora queda el problema de cómo decide el fabricante del navegador cuáles anclas de confianza propuestas son confiables y cuáles no. Queda a criterio del usuario confiar en el fabricante del navegador para elegir las mejores opciones y no simplemente aprobar todas las anclas de confianza, dispuesto a pagar sus cuotas de inclusión. La mayoría de los navegadores permiten que los usuarios inspeccionen las claves de la raíz (por lo general en la forma de certificados firmados por la raíz) y eliminen las que parezcan sospechosas.

Directorios

Otro problema de cualquier PKI es en dónde están almacenados los certificados (y sus cadenas de vuelta hacia un ancla de confianza conocida). Una posibilidad es hacer que cada usuario almacene sus propios certificados. Si bien esto es seguro (es decir, no hay forma de que los usuarios falsifiquen certificados firmados sin que esto se detecte), a la vez es inconveniente. Una alternativa que se ha propuesto es utilizar el

DNS como un directorio de certificados. Antes de contactar a Bob, es probable que Alice tenga que buscar la dirección IP de Bob mediante DNS, así que, ¿por qué no hacer que el DNS devuelva toda la cadena de certificados de Bob, junto con su dirección IP?

Algunas personas piensan que ésta es la forma de hacer las cosas, pero tal vez otras prefieran servidores de directorios dedicados cuyo único trabajo sea manejar los certificados X.509. Tales directorios podrían proporcionar servicios de búsqueda mediante el uso de las propiedades de los nombres X.500. Por ejemplo, en teoría un servicio de directorio como éste podría contestar una consulta como: "Dame una lista de todas las personas que tengan el nombre Alice y que trabajen en los departamentos de ventas en cualquier lugar de Estados Unidos o Canadá".

Revocación

El mundo real también está lleno de certificados, como los pasaportes y las licencias de conducir. Algunas veces estos certificados pueden revocarse; por ejemplo, las licencias de conducir pueden revocarse por conducir en estado de ebriedad y por otros delitos de manejo. En el mundo digital ocurre el mismo problema: el otorgante de un certificado podría decidir revocarlo porque la persona u organización que lo posee ha abusado de él en cierta manera. También puede revocarse si la clave privada del sujeto se ha expuesto o, peor aún, si la clave privada de la CA ha sido comprometida. Por lo tanto, una PKI necesita lidiar con el problema de la revocación. La posibilidad de la revocación complica las cosas.

Un primer paso en esta dirección es hacer que cada CA emita en forma periódica una **CRL** (**Lista de Revocación de Certificados**, del inglés *Certificate Revocation List*) que proporcione los números seriales de todos los certificados que ha revocado. Puesto que los certificados contienen tiempos de expiración, la CRL sólo necesita contener los números seriales de los certificados que no han expirado todavía. Una vez que pasa el tiempo de expiración de un certificado, éste se invalida de manera automática, por lo que no hay necesidad de hacer una distinción entre los certificados que han expirado y los que fueron revocados. En ninguno de los casos se puede usar uno de esos certificados.

Por desgracia, introducir listas CRL significa que un usuario que está próximo a utilizar un certificado debe adquirir la CRL para ver si éste se revocó. Si es así, dicho certificado no debe utilizarse. Sin embargo, si el certificado no está en la lista, pudo haber sido revocado justo después de que se publicó la lista. Por lo tanto, la única manera de estar seguro realmente es preguntar a la CA. Y la siguiente vez que se utilice ese mismo certificado, se le tiene que preguntar de nuevo a la CA, puesto que pudo haber sido revocado segundos antes.

Otra complicación es que un certificado revocado puede reinstalarse nuevamente; por ejemplo, si se revocó por no pagar una cuota que ya se encuentra al corriente. Al tener que lidiar con la revocación (y quizá con la reinstalación), se elimina una de la mejores propiedades de los certificados; es decir, que pueden utilizarse sin tener que contactar a una CA.

¿Dónde deben almacenarse las listas CRL? Un buen lugar sería el mismo en el que se almacenan los certificados. Una estrategia es que una CA quite de manera activa y periódica listas CRL y hacer que los directorios las procesen con sólo eliminar los certificados revocados. Si no se utilizan directorios para almacenar certificados, las listas CRL pueden almacenarse en caché en varias ubicaciones alrededor de la red. Puesto que una CRL es por sí misma un documento firmado, si se altera, esa alteración puede detectarse con facilidad.

Si los certificados tienen tiempos de vida largos, las listas CRL también los tendrán. Por ejemplo, si las tarjetas de crédito son válidas durante 5 años, el número de revocaciones pendientes será mucho más grande que si se emitieran nuevas tarjetas cada 3 meses. Una forma estándar para tratar con grandes listas CRL es emitir algunas veces una lista maestra, pero emitir actualizaciones con más frecuencia. Hacer esto reduce el ancho de banda necesario para distribuir las listas CRL.

8.6 SEGURIDAD EN LA COMUNICACIÓN

Hemos terminado nuestro estudio de las herramientas en cuestión. Ya cubrimos la mayoría de las técnicas y protocolos importantes. El resto del capítulo trata sobre cómo se utilizan estas técnicas en la práctica para proporcionar seguridad de red, junto con algunas reflexiones acerca de los aspectos sociales de la seguridad al final del capítulo.

En las siguientes cuatro secciones veremos la seguridad en la comunicación; es decir, cómo llevar los bits de manera secreta y sin modificación desde el origen hasta el destino y cómo mantener fuera a los bits no deseados. Éstos no son de ningún modo los únicos aspectos de seguridad en las redes, pero sin duda están entre los más importantes, lo que hace de éste un buen lugar para comenzar nuestro análisis.

8.6.1 IPsec

La IETF ha sabido por años que hay una escasez de seguridad en Internet. Agregarla no era fácil pues surgió una controversia acerca de dónde colocarla. La mayoría de los expertos en seguridad creían que para ofrecer una verdadera seguridad, el sistema de cifrado y las verificaciones de integridad tenían que llevarse a cabo de extremo a extremo (en la capa de aplicación). Esto es, el proceso de origen encripta o protege la integridad de los datos y los envía al proceso de destino, en donde se desencriptan o verifican. Por lo tanto, es posible detectar cualquier alteración que se realice entre estos dos procesos, o en cualquier sistema operativo. El problema con este enfoque es que requiere cambiar todas las aplicaciones para que estén conscientes de la seguridad. Desde esta perspectiva, el siguiente enfoque más conveniente es colocar el encriptado en la capa de transporte o en una nueva capa entre la capa de aplicación y la de transporte, con lo que se conserva el enfoque de extremo a extremo pero no hay que cambiar las aplicaciones.

La perspectiva opuesta es que los usuarios no entiendan la seguridad y no sean capaces de utilizarla de manera correcta, así como que nadie desee modificar los programas existentes de ninguna forma, por lo que la capa de red debe autentificar o encriptar paquetes sin que los usuarios estén involucrados. Después de años de batallas encarnizadas, esta perspectiva ganó suficiente soporte como para definir un estándar de seguridad de capa de red. En parte, el argumento era que al tener el encriptado en la capa de red no se evitaba que los usuarios conscientes de la seguridad hicieran lo correcto, además de que en cierto punto ayuda a los usuarios no conscientes de la seguridad.

El resultado de esta guerra fue un diseño llamado **IPsec** (**Seguridad IP**, del inglés *IP security*), el cual se describe en los RFC 2401, 2402 y 2406, entre otros. No todos los usuarios desean encriptado (ya que exige muchos recursos computacionales); por esta razón, en lugar de hacerlo opcional, se decidió requerir siempre pero permitir el uso de un algoritmo nulo. Este algoritmo nulo se describe y alaba por su simplicidad, facilidad de implementación y gran velocidad en el RFC 2410.

El diseño IPsec completo es un marco de trabajo para múltiples servicios, algoritmos y niveles de granularidad. La razón de los servicios múltiples es que no todas las personas quieren pagar el precio por tener todos los servicios todo el tiempo, por lo que están disponibles en todo momento. Los servicios principales son confidencialidad, integridad de datos y protección contra ataques de repetición (en donde un intruso repite una conversación). Todos estos servicios se basan en criptografía de clave simétrica debido a que es imprescindible un alto desempeño.

La razón de tener múltiples algoritmos es que un algoritmo que en estos momentos se considera seguro, se puede quebrantar en el futuro. Al hacer que el algoritmo IPsec sea independiente, el marco de trabajo puede sobrevivir incluso si alguno específico se quebranta más tarde.

La razón de tener múltiples niveles de granularidad es para que sea posible proteger una sola conexión TCP, todo el tráfico entre un par de hosts o todo el tráfico entre un par de enrutadores seguros, entre otras posibilidades.

Un aspecto poco sorprendente de IPsec es que, aun cuando se encuentra en la capa IP, es orientado a conexión. En la actualidad, eso no es tan sorprendente porque para tener seguridad, se debe establecer y utilizar una clave durante cierto periodo (en esencia, un tipo de conexión mediante un nombre distinto). Además, las conexiones amortizan los costos de configuración sobre muchos paquetes. Una "conexión" en el contexto de IPsec se conoce como **SA** (**Asociación de Seguridad**, del inglés *Security Association*). Una SA es una conexión símple entre dos puntos finales y tiene un identificador de seguridad asociado con ella. Si se necesita tráfico seguro en ambas direcciones, se requieren dos asociaciones de seguridad. Los identificadores de seguridad se transportan en paquetes que viajan en estas conexiones seguras, y se utilizan para buscar claves además de otra información relevante cuando llega un paquete seguro.

Técnicamente, IPsec tiene dos partes principales. La primera describe dos encabezados nuevos que se pueden agregar a los paquetes para transportar el identificador de seguridad, los datos de control de integridad y demás información. La otra parte, **ISAKMP** (**Asociación para Seguridad en Internet y Protocolo de Administración de Claves**, del inglés *Internet Security Association and Key Management Protocol*), se encarga de establecer las claves. ISAKMP es un marco de trabajo. El protocolo principal que realiza el trabajo es **IKE** (**Intercambio de Claves de Internet**, del inglés *Internet Key Exchange*). Hay que usar la versión 2 de **IKE** según lo descrito en el RFC 4306, ya que la versión anterior estaba plagada de fallas, como se indica en Perlman y Kaufman (2000).

IPsec puede utilizarse en uno de dos modos. En el **modo de transporte**, el encabezado IPsec se inserta justo después del encabezado IP. El campo *Protocolo* del encabezado IP se modifica para indicar que sigue un encabezado IPsec después del encabezado IP normal (antes del encabezado TCP). El encabezado IPsec contiene información de seguridad, principalmente el identificador SA, un nuevo número de secuencia y tal vez una verificación de integridad del campo de carga.

En el **modo de túnel**, todo el paquete IP, con encabezado y demás información, se encapsula en el cuerpo de un paquete IP nuevo con un encabezado IP totalmente nuevo. El modo de túnel es útil cuando termina en una ubicación que no sea el destino final. En algunos casos, el final del túnel es una máquina de puerta de enlace de seguridad; por ejemplo, el *firewall* de una empresa. Éste es el caso común para una VPN (Red Privada Virtual). En este modo, la puerta de enlace de seguridad encapsula y desencapsula paquetes conforme pasan a través de ella. Al terminar el túnel en esta máquina segura, las máquinas en la LAN de la empresa no tienen que estar al tanto de IPsec. Sólo la puerta de enlace de seguridad tiene que saber acerca de ello.

El modo de túnel también es útil cuando se agrega un conjunto de conexiones TCP y se maneja como un solo flujo cifrado, porque así se evita que un intruso vea quién está enviando cuántos paquetes a quién. Algunas veces el simple hecho de saber cuánto tráfico está pasando y hacia dónde se dirige es información valiosa. Por ejemplo, si durante una crisis militar, la cantidad de tráfico que fluye entre el Pentágono y la Casa Blanca se reduce de manera considerable, pero la cantidad de tráfico entre el Pentágono y alguna instalación militar oculta entre las Montañas Rocosas de Colorado se incrementa en la misma proporción, un intruso podría ser capaz de deducir alguna información útil a partir de estos datos. El estudio de los patrones de flujo de los paquetes, aunque estén encriptados, se conoce como **análisis de tráfico**. El modo de túnel proporciona una forma de frustrarlo hasta cierto punto. La desventaja del modo de túnel es que agrega un encabezado IP adicional, por lo que se incrementa el tamaño del paquete en forma considerable. En contraste, el modo de transporte no afecta tanto al tamaño del paquete.

El primer nuevo encabezado es **AH** (**Encabezado de Autentificación**, del inglés *Authentication Header*), el cual proporciona la verificación de integridad y la seguridad antirrepetición, pero no la confidencialidad (es decir, no hay encriptación de datos). En la figura 8-27 se ilustra el uso de AH en el modo de transporte. En el IPv4 se interpone entre los encabezados IP (incluyendo todas las opciones) y TCP. En IPv6 es sólo otro encabezado de extensión y se trata como tal. De hecho, el formato es parecido al de un encabezado de extensión IPv6 estándar. Tal vez haya que rellenar la carga útil hasta cierta longitud específica para el algoritmo de autentificación, como se muestra a continuación.

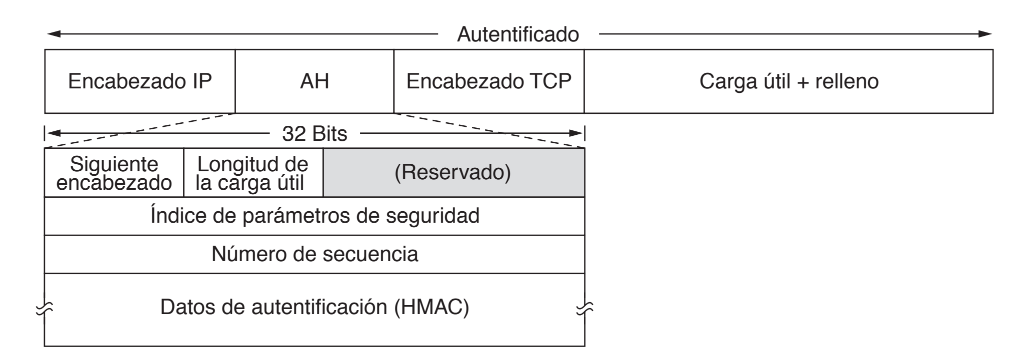


Figura 8-27. El encabezado de autentificación de IPsec en el modo de transporte para IPv4.

Examinemos ahora el encabezado AH. El campo *Siguiente encabezado* se utiliza para almacenar el valor que tenía el campo *Protocolo* de IP antes de reemplazarlo con 51 para indicar que sigue un encabezado AH. En muchos casos, aquí irá el código para TCP (6). La *Longitud de carga útil* es el número de palabras de 32 bits en el encabezado AH menos 2.

El *Índice de parámetros de seguridad* es el identificador de la conexión. El emisor lo inserta para indicar un registro específico en la base de datos del receptor. Este registro contiene la clave compartida que se utiliza en esta conexión y demás información sobre ella. Si la ITU hubiera inventado este protocolo en vez de la IETF, este campo se hubiera llamado *Número de circuitos virtuales*.

El campo *Número de secuencia* se utiliza para numerar todos los paquetes enviados en una SA. Cada paquete recibe un número único, incluso las retransmisiones. En otras palabras, la retransmisión de un paquete obtiene aquí un número diferente al del paquete original (aunque su número de secuencia TCP sea el mismo). El propósito de este campo es detectar ataques de repetición. Tal vez estos números de secuencia no se ajusten. Si se agotan todos los 2^{32} números, es necesario establecer una nueva SA para continuar la comunicación.

Por último tenemos el campo *Datos de autentificación*, que es de longitud variable y contiene la firma digital de la carga útil. Cuando se establece la SA, los dos lados negocian cuál algoritmo de firmas van a utilizar. Por lo general, aquí no se utiliza la criptografía de clave pública porque los paquetes se deben procesar con extrema rapidez y todos los algoritmos de clave pública conocidos son muy lentos. Puesto que IPsec se basa en la criptografía de clave simétrica y el emisor negocia con el receptor una clave compartida antes de establecer una SA, dicha clave compartida se utiliza en el cálculo de la firma. Una forma simple es calcular el hash sobre el paquete más la clave compartida. Desde luego que esta clave no se transmite. Un esquema como éste se conoce como **HMAC** (**Código de Autentificación de Mensajes Basado en Hash**, del inglés *Hashed Message Authentication Code*). Es mucho más rápido realizar el cálculo que primero ejecutar el SHA-1 y luego el RSA sobre el resultado.

El encabezado AH no permite la encriptación de los datos, por lo que es más útil cuando se necesita la verificación de la integridad pero no la confidencialidad. Una característica de AH que vale la pena mencionar es que la verificación de integridad abarca algunos de los campos en el encabezado IP; es decir, aquellos que no cambian conforme el paquete se mueve de un enrutador a otro. Por ejemplo, el campo *Tiempo de vida* cambia en cada salto, de manera que no se puede incluir en la verificación de integridad. Sin embargo, la dirección IP de origen se incluye en la verificación, lo que hace imposible que un intruso falsifique el origen de un paquete.

El encabezado IPsec alternativo es **ESP** (**Carga útil de Encapsulamiento de Seguridad**, del inglés *Encapsulating Security Payload*). En la figura 8-28 se muestra su uso para modo de transporte y para modo de túnel.

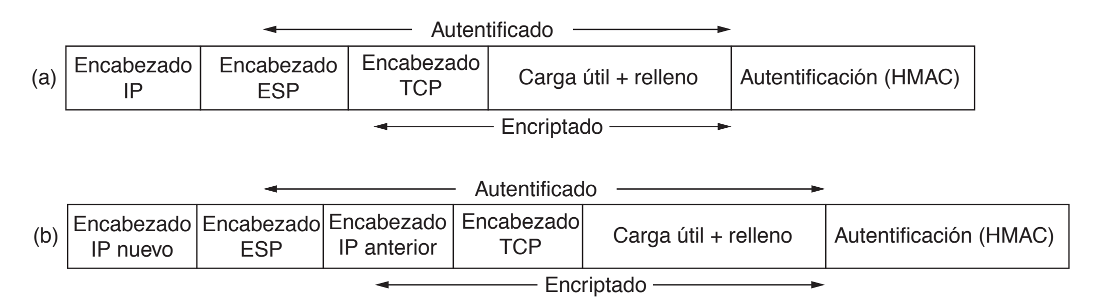


Figura 8-28. (a) ESP en modo de transporte. (b) ESP en modo de túnel.

El encabezado ESP está constituido por dos palabras de 32 bits. Éstas son los campos *Índice de parámetros de seguridad* y *Número de secuencia* que vimos en AH. Una tercera palabra que por lo general va después de ellos (pero que técnicamente no es parte del encabezado) es el *Vector de inicialización* que se utiliza para la encriptación de datos, a menos que se utilice la encriptación nula, en cuyo caso se omite.

ESP también incluye verificaciones de integridad HMAC, al igual que AH, pero en lugar de incluirse en el encabezado, van después de la carga útil como se muestra en la figura 8-28. Colocar el HMAC al final tiene una ventaja en una implementación de hardware: el HMAC se puede calcular conforme los bits se transmiten por la interfaz de red y se agregan al final. Ésta es la razón por la cual Ethernet y otras LANs tienen sus códigos CRC en un terminador, en lugar de en un encabezado. Con AH, el paquete tiene que almacenarse en el búfer y la firma tiene que calcularse antes de que se envíe el paquete, con lo que se reduce de manera potencial el número de paquetes/seg que pueden enviarse.

Dado que ESP puede hacer lo mismo que AH y más, y debido a que es más eficiente en el arranque, surge la pregunta: ¿por qué molestarse en tener a AH? La respuesta es en su mayor parte por cuestiones históricas. En un principio, AH se encargaba sólo de la integridad y ESP sólo de la confidencialidad. Más tarde se agregó la integridad a ESP, pero las personas que diseñaron AH no querían dejarlo morir después de todo el trabajo que habían realizado. Su único argumento válido es que AH verifica parte del encabezado IP, lo cual ESP no hace, pero es un argumento débil. Otro argumento débil es que un producto que soporta AH y no ESP podría tener menos problemas para obtener una licencia de exportación, porque no puede realizar la encriptación. Es probable que AH sea desplazado en el futuro.

8.6.2 *Firewalls*

La capacidad de conectar una computadora en cualquier lugar, con cualquier otra computadora de cualquier lugar, es una ventaja a medias. Para los usuarios domésticos navegar en Internet representa mucha diversión. Para los gerentes de seguridad empresarial es una pesadilla. Muchas empresas tienen grandes cantidades de información confidencial en línea secretos comerciales, planes de desarrollo de productos, estrategias de marketing, análisis financieros, etc. Si esta información cayera en manos de un competidor, las consecuencias podrían ser devastadoras.

Además del peligro de la fuga de información, también existe el peligro de la infiltración de información. En particular, los virus, gusanos y otras pestes digitales pueden abrir brechas de seguridad, destruir datos valiosos y hacer que los administradores pierdan mucho tiempo tratando de arreglar el daño que hayan hecho. Por lo general, se importan debido a los empleados descuidados que desean ejecutar algún nuevo juego ingenioso.

En consecuencia, se necesitan mecanismos para mantener los bits “buenos” dentro y los bits “malos” fuera. Un método es utilizar IPsec. Este método protege a los datos en tránsito entre los sitios seguros.

Sin embargo, IPsec no hace nada por proteger a la LAN de la compañía contra las pestes digitales y los intrusos. Para saber cómo alcanzar ese objetivo, necesitamos dar un vistazo a los *firewalls*.

Los ***firewalls*** (**servidores de seguridad**) son simplemente una adaptación moderna de la vieja estrategia medieval de seguridad: excavar un foso defensivo profundo alrededor de su castillo. Este diseño obligaba a que todos los que entraran o salieran del castillo pasaran a través de un único puente levadizo, en donde los encargados de la E/S los pudieran inspeccionar. En las redes es posible el mismo truco: una compañía puede tener muchas redes LAN conectadas de forma arbitraria, pero todo el tráfico que entra y sale de la compañía debe pasar a través de un puente levadizo electrónico (*firewall*), como se muestra en la figura 8-29. No existe ninguna otra ruta.

El *firewall* actúa como un **filtro de paquetes**. Inspecciona todos y cada uno de los paquetes entrantes y salientes. Los paquetes que cumplen cierto criterio descrito en reglas formuladas por el administrador de la red se reenvían en forma normal. Los que fallan la prueba simplemente se descartan.

Por lo general, los criterios de filtrado se proporcionan como reglas o tablas que listan los orígenes y destinos aceptables, los orígenes y destinos bloqueados y las reglas predeterminadas acerca de lo que se debe hacer con los paquetes que entran y salen a otras máquinas. En el caso común de una configuración TCP/IP, un origen o destino podría consistir en una dirección IP y un puerto. Los puertos indican el servicio deseado. Por ejemplo, el puerto TCP 25 es para el correo y el puerto TCP 80 es para HTTP. Algunos puertos simplemente pueden estar bloqueados. Por ejemplo, una empresa podría bloquear los paquetes entrantes para todas las direcciones IP combinadas con el puerto TCP 79. Una vez fue popular para el servicio Finger, el cual buscaba las direcciones de correo electrónico de las personas, pero se utiliza muy poco en la actualidad.

Otros puertos no se bloquean tan fácilmente. La dificultad es que los administradores de red desean seguridad, pero no pueden cortar la comunicación con el mundo exterior. Ese arreglo sería mucho más simple y eficiente para la seguridad, pero las quejas de los usuarios finales no terminarían nunca. Aquí es donde pueden ser útiles los arreglos como la **DMZ** (**Zona Desmilitarizada**, del inglés *DeMilitarized Zone*), que se muestra en la figura 8-29. La DMZ es la parte de la red de la empresa que se encuentra fuera del perímetro de seguridad. Cualquier cosa puede pasar aquí. Al colocar una máquina tal como un servidor web en la DMZ, las computadoras en Internet se pueden comunicar con ella para navegar por el sitio web de la empresa. Ahora el *firewall* se puede configurar para bloquear el tráfico TCP entrante al puerto 80, de modo que las computadoras en Internet no puedan usar este puerto para atacar a las computadoras en la red interna. Para poder administrar el servidor web, el *firewall* puede tener una regla que permita conexiones entre las máquinas internas y el servidor web.

Con el tiempo, los *firewall* se han vuelto mucho más sofisticados en una carrera armamentista contra los atacantes. En un principio, los *firewall* aplicaban una regla que se establecía de manera independiente

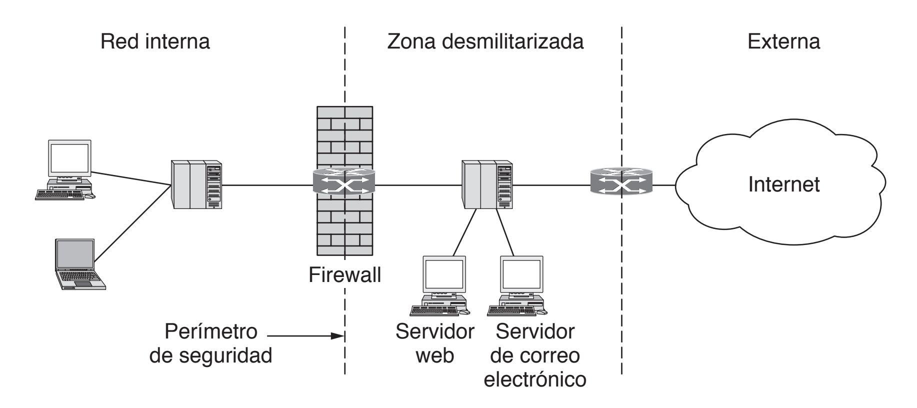


Figura 8-29. Un *firewall* que protege a una red interna.

para cada paquete, pero se dificultaba el proceso de escribir reglas que permitieran una funcionalidad útil y bloquearan a la vez todo el tráfico no deseado. Los ***firewall* con estado** asocian paquetes a las conexiones y usan campos del encabezado TCP/IP para mantener un registro de las conexiones. Esto permite usar reglas que, por ejemplo, permitan a un servidor web externo enviar paquetes a un host interno, pero sólo si ese host interno establece primero una conexión con el servidor web externo. Dicha regla no es posible con diseños sin estado, en los que se deben pasar o descartar todos los paquetes del servidor web externo.

Otro nivel de sofisticación superior al procesamiento con estado es que el *firewall* implemente **puertas de enlace a nivel de aplicación**. Este procesamiento implica que el *firewall* analice el interior de los paquetes, todavía más allá del encabezado TCP, para ver lo que está haciendo la aplicación. Con esta capacidad es posible diferenciar el tráfico HTTP que se utiliza para navegar por la web del tráfico HTTP utilizado para compartir archivos entre iguales. Los administradores pueden escribir reglas para evitar la compartición de archivos de igual a igual dentro de la empresa, pero permitir la navegación web que es vital para los negocios. En todos estos métodos se puede inspeccionar tanto el tráfico de salida como el de entrada; por ejemplo, para evitar que documentos confidenciales se envíen por correo electrónico fuera de la empresa.

La discusión anterior deja en claro que los *firewall* violan la distribución estándar en capas de los protocolos. Son dispositivos de capa de red, pero husmean en las capas de transporte y de aplicación para realizar su operación de filtrado. Esto los hace frágiles. Por ejemplo, los *firewall* tienden a depender de las convenciones de numeración de los puertos para determinar el tipo de tráfico que transporta un paquete. A menudo se utilizan los puertos estándar, pero no todas las computadoras ni todas las aplicaciones lo hacen. Algunas aplicaciones de igual a igual seleccionan puertos en forma dinámica para evitar que las detecten (y bloqueen) fácilmente. La encriptación mediante IPSEC u otros esquemas oculta la información de las capas superiores al *firewall*. Por último, un *firewall* no puede comunicarse fácilmente con las computadoras que se comunican a través de él para decirles qué políticas se están aplicando y por qué se va a cortar su conexión. Simplemente debe pretender ser un cable roto. Por todas estas razones, los puristas de las redes consideran que los *firewall* son una imperfección en la arquitectura de Internet. Sin embargo, Internet puede ser un lugar peligroso si usted es una computadora. Los *firewall* ayudan con ese problema, por lo que es probable que se sigan utilizando.

Incluso aunque el *firewall* esté configurado a la perfección, de todas formas existirán muchos problemas de seguridad. Por ejemplo, si un *firewall* está configurado para permitir sólo paquetes entrantes de redes específicas (por ejemplo, las demás plantas de la empresa), un intruso afuera del *firewall* puede introducir falsas direcciones de origen para evadir esta verificación. Si un miembro interno desea enviar documentos secretos, puede encriptarlos o incluso fotografiarlos y enviar las fotos como archivos JPEG, los cuales pueden evadir cualquier filtro de correo electrónico. Y no hemos analizado todavía el hecho de que, aunque tres cuartas partes de los ataques provienen del exterior del *firewall*, por lo general los ataques más dañinos son los que provienen del interior; por ejemplo, de empleados descontentos (Verizon, 2009).

Hay un problema distinto con los *firewall*: proporcionan un solo perímetro de defensa. Si se viola esa defensa, se cierran todas las apuestas. Por esta razón, los *firewall* se utilizan comúnmente en una defensa por capas. Por ejemplo, un *firewall* puede proteger la entrada a la red interna y cada computadora puede ejecutar también su propio *firewall*. Sin duda, los lectores que piensen que un solo punto de verificación de seguridad es suficiente no han tomado recientemente un vuelo internacional en una aerolínea programada.

Además, hay otra clase de ataques que los *firewall* no pueden manejar. La idea básica de un *firewall* es evitar que entren intrusos y que salga información secreta. Por desgracia, hay personas que no tienen nada mejor que hacer que tratar de inhabilitar ciertos sitios. Para ello envían grandes cantidades de paquetes legítimos al destino, hasta que el sitio se colapsa debido a la carga. Por ejemplo, para derribar un sitio web, un intruso puede enviar un paquete TCP *SYN* para establecer una conexión. A continuación el

sitio asignará una ranura en la tabla para la conexión y enviará como respuesta un paquete *SYN* + *ACK*. Si el intruso no responde, la ranura de la tabla se conservará durante algunos segundos hasta que expire. Si el intruso envía miles de solicitudes de conexión, todas las ranuras de la tabla se llenarán y no será posible establecer ninguna conexión legítima. Los ataques en los que el objetivo del intruso es bloquear el destino en lugar de robar datos se conocen como ataques **DoS** (**Negación de Servicio**, del inglés *Denial of Service*). Por lo general, los paquetes de solicitud tienen direcciones de origen falsas, por lo que no es fácil rastrear al intruso. Los ataques DoS contra grandes sitios web son comunes en Internet.

Una variante aún peor es aquella en la que el intruso ha entrado en cientos de computadoras en cualquier parte del mundo, y después ordena a todas ellas que ataquen al mismo objetivo y al mismo tiempo. Este método no sólo incrementa el poder del intruso, sino que también reduce la probabilidad de detectarlo debido a que los paquetes provienen de una gran cantidad de máquinas que pertenecen a usuarios ingenuos. Un ataque de este tipo se conoce como ataque **DDoS** (**Negación de Servicio Distribuida**, del inglés *Distributed Denial of Service*). Es difícil defenderse de un ataque como éste. Incluso si la máquina atacada puede reconocer rápidamente una solicitud falsa, le toma algún tiempo procesar y descartar la solicitud, y si llegan suficientes solicitudes por segundo, la CPU pasará todo su tiempo ocupándose de ellas.

8.6.3 Redes privadas virtuales

Muchas empresas tienen oficinas y plantas esparcidas en muchas ciudades, algunas veces hasta en varios países. En el pasado, antes de que existieran las redes de datos públicas, era común que algunas empresas alquilaran líneas a las compañías telefónicas para todas o sólo algunas ubicaciones. Algunas empresas aún hacen esto. Una red constituida por computadoras de la empresa y líneas telefónicas alquiladas se conoce como **red privada**.

Las redes privadas funcionan bien y son muy seguras. Si las únicas líneas disponibles son las alquiladas, el tráfico no puede fugarse de las ubicaciones de la empresa y los intrusos tienen que intervenir físicamente las líneas para infiltrarse, lo cual no es fácil de hacer. El problema con las redes privadas es que alquilar una sola línea T1 entre dos puntos cuesta miles de dólares mensuales, y las líneas T3 son mucho más costosas. Cuando aparecieron las redes de datos públicas y después Internet, muchas empresas quisieron trasladar su tráfico de datos (y posiblemente de voz) a la red pública, aunque sin renunciar a la seguridad de la red privada.

Esta demanda pronto llevó a la invención de las redes **VPN** (**Redes Privadas Virtuales**, del inglés *Virtual Private Networks*), que son redes superpuestas sobre redes públicas, pero con muchas propiedades de las redes privadas. Se llaman "virtuales" porque son sólo una ilusión, al igual que los circuitos virtuales no son reales ni la memoria virtual es real.

Un método popular es construir redes VPN directamente sobre Internet. Un diseño común es equipar cada oficina con un *firewall* y crear túneles a través de Internet entre todos los pares de oficinas, como se ilustra en la figura 8-30(a). Una ventaja adicional de usar Internet para la conectividad es que los túneles se pueden establecer bajo demanda para incluir, por ejemplo, la computadora de un empleado que se encuentra en su hogar o de viaje, siempre y cuando tenga conexión a Internet. Esta flexibilidad es mucho mayor de la que proveen las líneas alquiladas sin embargo, desde la perspectiva de las computadoras en la VPN, la topología es igual a la del caso de la red privada, como se muestra en la figura 8-30(b). Cuando se inicia el sistema, cada par de *firewall* tiene que negociar los parámetros de su SA, incluyendo los servicios, modos, algoritmos y claves. Si se utiliza IPsec para los túneles, es posible agregar todo el tráfico entre cualquier par de oficinas en una sola SA autentificada y encriptada, con lo cual se obtiene el control de la integridad, la confidencialidad e incluso una inmunidad considerable al análisis de tráfico. Muchos *firewall* tienen capacidades VPN integradas. Algunos enrutadores ordinarios también pueden

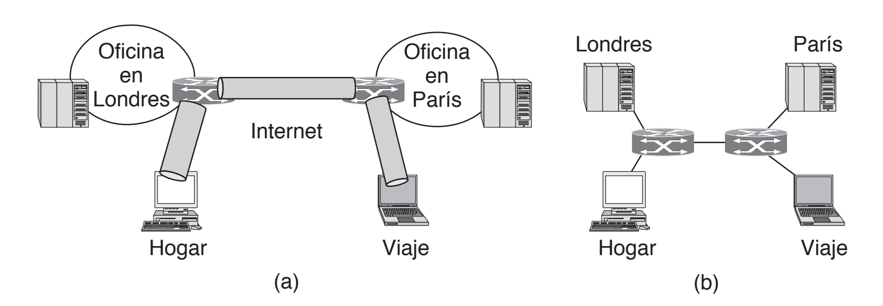


Figura 8-30. (a) Una red privada virtual. (b) La topología vista desde el interior.

hacer esto, pero debido a que los firewalls están principalmente en el negocio de la seguridad, es natural que los túneles empiecen y terminen en los *firewall*, con lo cual se establece una clara separación entre la empresa e Internet. Por lo tanto, los *firewall*, las redes VPN e IPsec con ESP en modo de túnel son una combinación natural y se utilizan mucho en la práctica.

Una vez que se han establecido las asociaciones SA, el tráfico puede comenzar a fluir. Para un enrutador en Internet, un paquete que viaja a través de un túnel VPN es sólo un paquete ordinario. Lo único extraño es la presencia del encabezado IPsec después del encabezado IP, pero debido a que estos encabezados adicionales no tienen efecto en el proceso de reenvío, los enrutadores no se preocupan por ellos.

Otro método que está ganando popularidad es hacer que el ISP establezca la VPN. Mediante el uso de MPLS (como vimos en el capítulo 5), se pueden establecer las rutas para el tráfico de la VPN a través de la red del ISP entre las oficinas de la empresa. Estas rutas mantienen separado el tráfico de la VPN del resto del tráfico de Internet y se puede garantizar cierta cantidad de ancho de banda u otro tipo de calidad del servicio.

Una ventaja principal de una VPN es la de ser completamente transparente para todo el software de usuario. Los *firewall* configuran y manejan las asociaciones SA. La única persona que está consciente de esta configuración es el administrador del sistema, quien tiene que configurar y gestionar las puertas de enlace de seguridad, o el administrador del ISP que tiene que configurar las rutas de MPLS. Para todos los demás, es como tener de nuevo una red privada mediante una línea alquilada. Para mayor información acerca de las redes VPN, consulte a Lewis (2006).

8.6.4 Seguridad inalámbrica

Es muy fácil diseñar un sistema mediante el uso de redes VPN y *firewall* que sea lógica y completamente seguro pero eso, en la práctica, puede fallar. Esta situación puede ocurrir si algunas de las máquinas son inalámbricas y utilizan comunicación de radio, la cual pasa justo encima del firewall en ambas direcciones. Con frecuencia, el rango de las redes 802.11 es de algunos cientos de metros, por lo que cualquiera que desee espiar una compañía sólo tiene que introducirse en el estacionamiento para empleados por la mañana, dejar una computadora portátil habilitada para 802.11 en el automóvil para que grabe todo lo que oiga y seguir con sus tareas cotidianas. Al anochecer, el disco duro estará repleto de valiosos datos. En teoría, se supone que esta fuga no debería suceder. Pero, en teoría, las personas tampoco deberían robar bancos.

La mayor parte del problema de seguridad se puede remontar a los fabricantes de las estaciones base inalámbricas (puntos de acceso), quienes tratan de hacer que sus productos sean amigables para el usuario. Por lo general, si el usuario saca el dispositivo de la caja y lo conecta en el enchufe de la energía eléctrica, comienza a operar de inmediato —casi siempre sin seguridad—, revelando secretos a quienes estén dentro del alcance de radio. Si a continuación se conecta a una red Ethernet, de pronto todo el tráfico

de ésta aparecerá también en el estacionamiento. La tecnología inalámbrica es el sueño de todo espía: datos gratuitos sin tener que hacer nada. Por lo tanto, sobra decir que la seguridad es mucho más importante para los sistemas inalámbricos que para los cableados. En esta sección veremos algunas formas en que las redes inalámbricas manejan la seguridad. Podrá encontrar información adicional en Nichols y Lekkas (2002).

Seguridad del 802.11

Una parte del estándar 802.11, que en un principio se conocía como **802.11i**, establece un protocolo de seguridad en el nivel de enlace de datos para evitar que un nodo inalámbrico lea o interfiera con los mensajes enviados entre otro par de nodos inalámbricos. También se le conoce mediante el nombre comercial **WPA2** (**Acceso Protegido WiFi 2**, del inglés *WiFi Protected Access 2*). El WPA simple es un esquema interino que implementa un subconjunto del 802.11i. Hay que evitarlo a favor del WPA2.

En breve describiremos el 802.11i, pero primero debemos señalar que es un reemplazo para el esquema **WEP** (**Privacidad Equivalente a cableado**, del inglés *Wired Eqivalent Privacy*), la primera generación de protocolos de seguridad 802.11. El WEP fue diseñado por un comité de estándares de redes, lo cual representa un proceso completamente distinto a (por ejemplo) la forma en que el NIST seleccionó el diseño del AES. Los resultados fueron devastadores. ¿Qué era lo que estaba mal? Pues resulta que casi todo, desde la perspectiva de seguridad. Por ejemplo, para encriptar los datos confidenciales, el WEP les aplicaba un OR exclusivo con la salida de un sistema de cifrado de flujo. Por desgracia, los arreglos de claves débiles significaban que la salida se reutilizaba con frecuencia. Esto condujo a formas triviales de quebrantar su seguridad. Como otro ejemplo, la verificación de integridad se basaba en un CRC de 32 bits. Es un código eficiente para detectar errores de transmisión, pero no es un mecanismo criptográfico sólido para vencer a los atacantes.

Gracias a éstos y a otros errores de diseño, era muy fácil comprometer la seguridad de WEP. La primera demostración práctica de que WEP había sido vencido llegó cuando Adam Stubblefield era un interno en AT&T (Stubblefield y colaboradores, 2002). Él pudo codificar y probar un ataque descrito por Fluhrer y colaboradores (2001) en una semana, de la cual la mayor parte del tiempo lo invirtió en convencer a la gerencia de que le compraran una tarjeta WiFi para usarla en sus experimentos. En la actualidad, el software para quebrantar contraseñas WEP en menos de un minuto está disponible en forma gratuita, por lo que no se recomienda usar WEP en ningún caso. Aunque evita el acceso casual, no provee ninguna forma real de seguridad. El grupo 802.11 se reunió de emergencia cuando era obvio que WEP estaba seriamente comprometido. En junio de 2004 se produjo un estándar formal.

Ahora describiremos el 802.11i, que proporciona una verdadera seguridad si se configura y utiliza en forma apropiada. Hay dos escenarios comunes en los que se utiliza el esquema WPA2. El primero es en un entorno corporativo, en donde una empresa tiene un servidor de autentificación separado con una base de datos de nombres de usuario y contraseñas, la cual se puede usar para determinar si un cliente inalámbrico puede o no acceder a la Red. En este entorno, los clientes usan protocolos estándar para autentificarse con la Red. Los estándares principales son: **802.1X**, en el que el punto de acceso permite al cliente dialogar con el servidor de autentificación y observa el resultado, y **EAP** (**Protocolo de Autentificación Extensible**, del inlgés *Extensible Authentication Protocol*) (RFC 3748), el cual indica cómo deben interactuar el cliente y el servidor de autentificación. En realidad, EAP es un marco de trabajo y otros estándares definen los mensajes del protocolo. Sin embargo, no entraremos en los diversos detalles de este intercambio, puesto que no son muy relevantes para nuestro análisis general.

El segundo escenario es en un entorno doméstico, en donde no hay servidor de autentificación. En cambio, hay una sola contraseña compartida que los clientes utilizan para acceder a la red inalámbrica. Esta configuración es menos compleja que tener un servidor de autentificación, razón por la cual se utiliza en el hogar y en negocios pequeños, pero también es menos seguro. La principal diferencia es que,

con un servidor de autentificación, cada cliente recibe una clave para encriptar el tráfico de modo que los demás clientes no puedan verlo. Con una sola contraseña compartida se producen distintas claves para cada cliente, pero todos los clientes tienen la misma contraseña y pueden obtener las claves de los demás si lo desean.

Las claves que se utilizan para encriptar el tráfico se calculan como parte de un acuerdo de autentificación. El acuerdo ocurre justo después de que el cliente se asocia con una red inalámbrica y se autentifica con un servidor de autentificación, si hay uno. Al inicio del acuerdo, el cliente tiene la contraseña de red compartida o su contraseña para el servidor de autentificación. Esta contraseña se utiliza para obtener una clave maestra. Sin embargo, ésta no se utiliza directamente para encriptar paquetes. La práctica criptográfica estándar es obtener una clave de sesión por cada periodo de uso, cambiar la clave para distintas sesiones y evitar exponer la clave maestra para que otros la observen el menor tiempo posible. Ésta es la clave de sesión que se calcula en el acuerdo.

La clave de sesión se calcula mediante el acuerdo de cuatro paquetes que se muestra en la figura 8-31. Primero, el AP (punto de acceso) envía un número aleatorio como identificación. Los números aleatorios que se utilizan sólo una vez en protocolos de seguridad como éste se llaman **nonces**, lo cual es algo así como una contracción de "número utilizado sólo una vez". El cliente también elige su propio nonce. Usa los nonces, su dirección MAC y la del AP, junto con la clave maestra para calcular una clave de sesión, K_s. La clave de sesión se divide en porciones, cada una de las cuales se utiliza para distintos fines, pero hemos omitido este detalle. Ahora el cliente tiene claves de sesión, pero el AP no. Por lo tanto, el cliente envía su nonce al AP y éste realiza el mismo cálculo para obtener las mismas claves de sesión. Los nonces se pueden enviar sin necesidad de encriptarlos, ya que las claves no se pueden obtener a partir de ellos sin la información adicional secreta. El mensaje del cliente está protegido con una verificación de integridad conocida como **MIC** (**Verificación de Integridad de Mensaje**, del inglés *Message Integrity Check*), la cual está basada en la clave de sesión. El AP puede verificar que la MIC sea correcta y que, por ende, el mensaje realmente debe provenir del cliente, una vez que calcula las claves de sesión. MIC es simplemente otro nombre para un código de autentificación de mensaje, como en un HMAC. Es común que el término MIC se utilice más para los protocolos de red debido a la potencial confusión con las direcciones MAC (Control de Acceso al Medio).

En los últimos dos mensajes, el AP distribuye una clave de grupo K_G al cliente, y éste confirma la recepción del mensaje. Al recibir estos mensajes, el cliente puede verificar que el AP tenga las claves de sesión correctas, y viceversa. La clave de grupo se usa para la difusión y multidifusión de tráfico en la LAN 802.11. Como el resultado del acuerdo es que cada cliente tiene sus propias claves de encriptación,

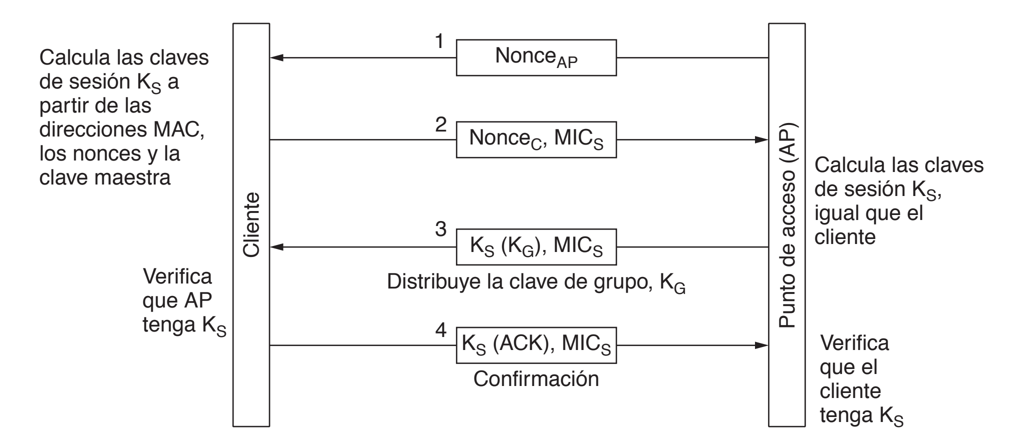


Figura 8-31. El acuerdo de establecimiento de claves de 802.11i.

el AP no puede usar ninguna de estas claves para difundir paquetes a todos los clientes inalámbricos; habría que enviar por separado una copia a cada cliente mediante el uso de su clave. En cambio, se distribuye una clave compartida para que el tráfico difundido se pueda enviar sólo una vez y lo puedan recibir todos los clientes. Se debe actualizar a medida que los clientes salen de la Red y se unen a ella.

Por último, llegamos a la parte en donde se utilizan las claves para proveer seguridad. Se pueden usar dos protocolos en el 802.11i para proveer confidencialidad en los mensajes, integridad y autentificación. Al igual que WPA, uno de los protocolos de nombre **TKIP** (**Protocolo de Integridad de Clave Temporal**, del inglés *Temporary Key Integrity Protocol*) era una solución provisional. Se diseñó para mejorar la seguridad en las tarjetas 802.11 antiguas y lentas, de modo que se pudiera proveer cuando menos un nivel mayor de seguridad al de WEP en una actualización del *firmware*. Sin embargo, también fue quebrantado, por lo que nos queda el otro protocolo recomendado, **CCMP**. ¿Qué significa CCMP? Es la abreviación del nombre un tanto espectacular: Protocolo de modo de contador con código de autentificación de mensaje mediante encadenamiento de bloques de sistema de cifrado. Nosotros lo llamaremos simplemente CCMP. Usted puede llamarlo como quiera.

El CCMP funciona de una manera bastante simple. Usa encriptación AES con una clave y tamaño de bloque de 128 bits. La clave proviene de la clave de sesión. Para proveer confidencialidad, los mensajes se encriptan con AES en modo de contador. Recuerde que en la sección 8.2.3 vimos los modos de sistema de cifrado. Estos modos son los que evitan que el mismo mensaje se encripte en cada ocasión con el mismo conjunto de bits. El modo de contador mezcla un contador con la encriptación. Para proveer integridad, el mensaje (incluyendo los campos de encabezado) se encripta con el modo de encadenamiento de bloques de sistema de cifrado, y se mantiene el último bloque de 128 bits como el MIC. Después se envían tanto el mensaje (encriptado con el modo de contador) y el MIC. El cliente y el AP pueden realizar cada uno esta encriptación, o pueden verificarla cuando se reciban un paquete inalámbrico. Para la difusión o multidifusión de mensajes, se utiliza el mismo procedimiento con la clave de grupo.

Seguridad de Bluetooth

Bluetooth tiene un rango mucho más corto que el 802.11, por lo que no es fácil atacarlo desde un estacionamiento, pero la seguridad sigue siendo un problema en este caso. Por ejemplo, imagine que la computadora de Alice está equipada con un teclado Bluetooth inalámbrico. Si no hubiera seguridad y Trudy se encontrara en la oficina adyacente, podría leer todo lo que Alice escribiera, incluyendo todo su correo electrónico saliente. También podría capturar todo lo que la computadora de Alice enviara a la impresora Bluetooth instalada junto a ella (por ejemplo, el correo electrónico entrante y los informes confidenciales). Por fortuna, Bluetooth dispone de un complejo esquema de seguridad para tratar de frustrar los ataques de todas las Trudy del mundo. Ahora sintetizaremos sus principales características.

Las versiones de Bluetooth 2.1 y posteriores tienen cuatro modos de seguridad, que van desde ninguna seguridad en absoluto hasta la encriptación completa de datos y el control de integridad. Al igual que con el 802.11, si se deshabilita la seguridad (la opción predeterminada para dispositivos antiguos), no hay seguridad. La mayoría de los usuarios mantiene deshabilitada la seguridad hasta que ocurre una infracción grave de seguridad; es entonces cuando la habilitan. En el mundo esta metodología se conoce como cerrar la puerta de la caballeriza después de que escapa el caballo.

Bluetooth proporciona seguridad en varias capas. En la capa física, los saltos de frecuencia proporcionan un poco de seguridad, pero debido a que es necesario indicar a cualquier dispositivo Bluetooth de una piconet la secuencia de saltos de frecuencia, es obvio que esta secuencia no es un secreto. La verdadera seguridad empieza cuando el esclavo recién llegado pide un canal al maestro. Antes de Bluetooth 2.1, se asumía que dos dispositivos compartían una clave secreta que se establecía por adelantado. En algunos casos, el fabricante es quien las incluye de manera fija (por ejemplo, unos auriculares y un teléfono móvil que se venden como una sola unidad). En otros casos, un dispositivo (por ejemplo, los auriculares)

tiene una clave integrada fija y el usuario tiene que introducirla en el otro dispositivo (por ejemplo, el teléfono móvil) como un número decimal. Estas claves compartidas se conocen como **claves de acceso** (*passkeys*). Por desgracia, es común que las claves de acceso se fijen en “1234” o cualquier otro valor predecible, y en cualquier caso son cuatro dígitos decimales, lo cual permite sólo 10^4 opciones. Con el emparejamiento simple seguro en Bluetooth 2.1, los dispositivos seleccionan un código de un rango de seis dígitos, lo cual hace que la clave de acceso sea menos predecible, pero de todas formas no es muy segura.

Para establecer un canal, tanto el esclavo como el maestro verifican si el otro conoce la clave de acceso. De ser así, negocian si ese canal será encriptado, si se controlará su integridad o ambas cosas. Después seleccionan una clave de sesión aleatoria de 128 bits, de las cuales algunas pueden ser públicas. El objetivo de permitir esta debilidad de clave es respetar las restricciones gubernamentales de varios países diseñadas para evitar la exportación o el uso de claves más grandes de las que el gobierno pueda quebrantar.

La encriptación utiliza un sistema de cifrado de flujo llamado $\mathbf{E_0}$; el control de integridad utiliza **SAFER+**. Ambos son sistemas de cifrado en bloque de clave simétrica tradicionales. SAFER+ participó en el concurso para el AES, pero se eliminó en la primera ronda porque era más lento que los otros candidatos. Bluetooth se terminó antes de que se eligiera el sistema de cifrado AES; de lo contrario quizá éste hubiera utilizado Rijndael.

En la figura 8-14 se muestra la encriptación actual que utiliza el sistema de cifrado de flujo, en donde se aplica un OR exclusivo al texto plano y al flujo de claves para generar el texto cifrado. Por desgracia, el mismo E_0 (al igual que RC4) puede tener debilidades fatales (Jakobsson y Wetzel, 2001). Si bien no han logrado quebrantarlo hasta el momento de escribir este libro, sus similitudes con el sistema de cifrado A5/1, cuya falla espectacular puso en peligro todo el tráfico telefónico de GSM, son causa de preocupación (Biryukov y colaboradores, 2000). Algunas veces la gente se sorprende (incluyendo a los autores de este libro) de que en el eterno juego del gato y el ratón entre los criptógrafos y los criptoanalistas, estos últimos salgan ganando con mucha frecuencia.

Otro problema de seguridad es que Bluetooth sólo autentica dispositivos y no usuarios, por lo que el robo de un dispositivo Bluetooth podría conceder al ladrón el acceso a la cuenta financiera y a otras cuentas del usuario. Sin embargo, Bluetooth también implementa seguridad en las capas superiores, por lo que incluso en el caso de una infracción de seguridad en el nivel de enlace, podría quedar algo de seguridad, en especial en las aplicaciones que requieren que se introduzca un código PIN en forma manual desde algún tipo de teclado para completar la transacción.

8.7 PROTOCOLOS DE AUTENTIFICACIÓN

La autentificación es la técnica mediante la cual un proceso verifica que su compañero de comunicación sea quien se supone que debe ser y no un impostor. Es muy difícil verificar la identidad de un proceso remoto en la presencia de un intruso activo y malicioso, además de que se requieren protocolos complejos basados en la criptografía. En esta sección estudiaremos algunos de los diversos protocolos de autentificación que se utilizan en redes de computadoras inseguras.

Vale la pena señalar que algunas personas confunden la autorización con la autentificación. Esta última tiene que ver con la interrogante de si usted realmente se está comunicando con un proceso específico. La autorización tiene que ver con lo que ese proceso tiene permitido hacer. Por ejemplo, suponga que un proceso cliente se contacta con un servidor de archivos y dice: “Soy el proceso de Scott y deseo eliminar el archivo *librococina.viejo*”. Desde el punto de vista del servidor de archivos, hay que responder a dos preguntas:

1. ¿Es éste realmente el proceso de Scott (autentificación)?
2. ¿Tiene Scott permitido eliminar *librococina.viejo* (autorización)?

Sólo después de contestar estas preguntas de manera afirmativa y sin ambigüedad, se puede realizar la acción solicitada. En realidad, la primera pregunta es la clave. Una vez que el servidor de archivos sabe con quién está hablando, verificar la autorización es sólo cuestión de buscar entradas en las tablas locales o en las bases de datos. Por esta razón, en esta sección nos concentraremos en la autentificación.

Éste es el modelo general que utilizan básicamente todos los protocolos de autentificación. Alice empieza por enviar un mensaje ya sea a Bob o a un **KDC** (**Centro de Distribución de Claves**, del inglés *Key Distribution Center*) de confianza, el cual se espera sea honesto. A continuación siguen otros intercambios de mensajes en varias direcciones. Conforme se envían estos mensajes, es probable que Trudy pueda interceptarlos, modificarlos o repetirlos para engañar a Alice y a Bob o simplemente para dañar el trabajo.

Sin embargo, una vez que se completa el protocolo, Alice está segura de que está hablando con Bob y él está seguro de que está hablando con ella. Asimismo, en la mayoría de los protocolos, ellos dos también habrán establecido una **clave de sesión secreta** para utilizarla en la próxima conversación. En la práctica, por cuestiones de desempeño, todo el tráfico de datos se encripta mediante el uso de criptografía de clave simétrica (por lo general AES o triple DES), aunque la criptografía de clave pública se utiliza mucho para los protocolos de autentificación mismos y para establecer la clave de sesión.

El objetivo de utilizar una nueva clave de sesión elegida al azar para cada nueva conexión es minimizar la cantidad de tráfico que se envía con las claves secretas o públicas del usuario, para reducir la cantidad de texto cifrado que un intruso puede obtener, y para minimizar el daño hecho si un proceso falla y su volcado de memoria cae en las manos equivocadas. Por fortuna, la única clave presente será entonces la de sesión. Hay que tener cuidado de poner en cero todas las claves permanentes después de que se haya establecido la sesión.

8.7.1 Autentificación basada en una clave secreta compartida

Para nuestro primer protocolo de autentificación daremos por hecho que Alice y Bob ya comparten una clave secreta, K_{AB}. Esta clave compartida podría haberse acordado por teléfono o en persona, pero de ninguna manera se debe acordar en la Red (insegura).

Este protocolo se basa en un principio que se encuentra en muchos protocolos de autentificación: una parte envía un número aleatorio a la otra, quien a continuación lo transforma de una manera especial y después regresa el resultado. Dichos protocolos se conocen como **protocolos de desafío-respuesta**. En éste y en los protocolos de autentificación subsecuente se utilizará la notación que se muestra a continuación:

A, B son las identidades de Alice y Bob.
R_i son los desafíos, en donde el subíndice i identifica al retador.
K_i son claves, en donde i indica al propietario.
K_S es la clave de sesión.

En la figura 8-32 se muestra la secuencia de mensajes de nuestro primer protocolo de autentificación de clave compartida. En el mensaje 1, Alice envía su identidad (A) a Bob en una forma que él entiende. Por supuesto que éste no tiene forma de saber si el mensaje proviene de Alice o de Trudy, por lo que elige un desafío, un número aleatorio grande R_B, y lo envía a "Alice" como mensaje 2, en texto plano. A continuación Alice encripta el mensaje con la clave que comparte con Bob y envía el texto cifrado, K_{AB} (R_B), de vuelta en el mensaje 3. Cuando Bob ve el mensaje sabe que proviene de Alice porque Trudy no conoce K_{AB} y, por lo tanto, no pudo haberlo generado. Además, puesto que R_B fue elegido de manera aleatoria a partir de un espacio grande (digamos, números aleatorios de 128 bits), no es probable que

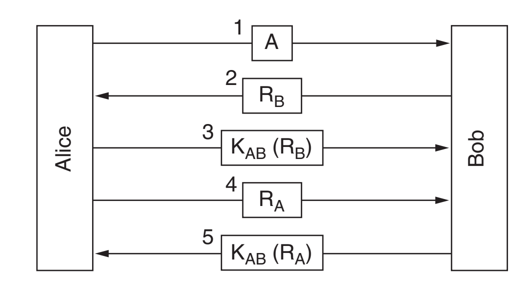


Figura 8-32. Autentificación de dos vías mediante el uso de un protocolo de desafío-respuesta.

Trudy haya visto a R_B y su respuesta en una sesión anterior. Tampoco es probable que pueda adivinar la respuesta correcta para cualquier desafío.

En este punto, Bob está seguro de que está hablando con Alice, pero ella no está segura de nada. Por lo que Alice sabe, Trudy podría haber interceptado el mensaje 1 y regresar R_B como respuesta. Tal vez Bob murió anoche. Para averiguar con quién está hablando, Alice elige un número al azar R_A y lo envía a Bob como texto plano, en el mensaje 4. Cuando Bob responde con K_{AB} (R_A), Alice sabe que está hablando con Bob. Si desean establecer una clave de sesión ahora, Alice puede elegir una (K_S), encriptarla con K_{AB} y enviarla a Bob.

El protocolo de la figura 8-32 contiene cinco mensajes. Veamos si podemos ser ingeniosos para eliminar algunos de ellos. En la figura 8-33 se muestra un método. Aquí Alice inicia el protocolo de desafío-respuesta en lugar de esperar a que Bob lo haga. De manera similar, mientras Bob responde al desafío de Alice, envía el suyo. Así, el protocolo se puede reducir a tres mensajes en lugar de cinco.

¿Es este nuevo protocolo una mejora del original? En cierto sentido sí: es más corto. Por desgracia, también es incorrecto. Bajo ciertas circunstancias, Trudy puede vencer este protocolo si utiliza lo que se conoce como un **ataque de reflexión**. En particular, Trudy puede romperlo, si es posible, para abrir múltiples sesiones con Bob al mismo tiempo. Por ejemplo, esta situación podría ocurrir si Bob es un banco y está preparado para aceptar muchas conexiones simultáneas de cajeros automáticos.

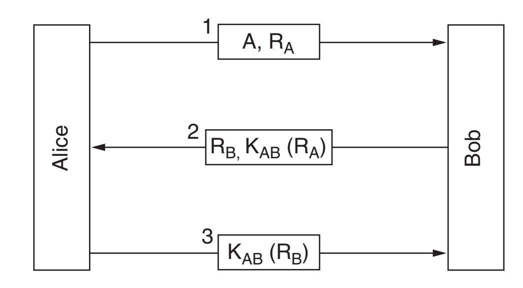


Figura 8-33. Un protocolo de autentificación de dos vías, abreviado.

En la figura 8-34 se muestra el ataque de reflexión de Trudy. Inicia cuando Trudy afirma que es Alice y envía R_T. Como siempre, Bob responde con su propio desafío, R_B. Ahora Trudy está atorada. ¿Qué puede hacer? No conoce K_{AB} (R_B).

Trudy puede abrir una segunda sesión con el mensaje 3 y proporcionar un R_B que tome del mensaje 2 como su desafío. Bob lo encripta con calma y regresa K_{AB} (R_B) en el mensaje 4. Sombreamos los mensajes de la segunda sesión para que resalten. Ahora Trudy tiene la información que le faltaba, por lo que puede completar la primera sesión y abortar la segunda. Bob ahora está convencido de que Trudy es Alice, por

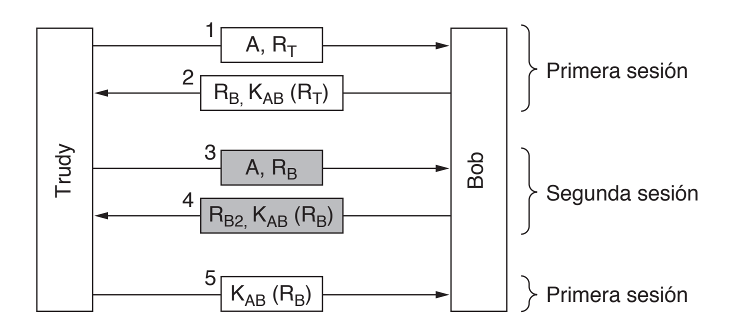


Figura 8-34. El ataque de reflexión.

lo que cuando ella le pide su estado de cuenta bancaria, Bob se lo proporciona sin más. Después, cuando ella le pide que transfiera todo su dinero a una cuenta secreta en un banco de Suiza, lo hace sin titubear.

La moraleja de esta historia es:

Diseñar un protocolo de autentificación correcto es más difícil de lo que parece.

A menudo, las siguientes cuatro reglas generales ayudan al diseñador a evitar los obstáculos comunes:

1. Haga que el iniciador demuestre que es quien dice ser antes de que el destinatario tenga que hacerlo. Esto evita que Bob revele información valiosa antes de que Trudy tenga que proporcionar evidencia de su identidad.
2. Haga que tanto el iniciador como el destinatario utilicen claves diferentes para probar, aunque esto signifique tener dos claves compartidas, K_{AB} y K'_{AB}.
3. Haga que el iniciador y el destinatario utilicen conjuntos diferentes para elaborar sus desafíos. Por ejemplo, el iniciador debe utilizar números pares y el destinatario números impares.
4. Haga que el protocolo resista a los ataques que involucren una segunda sesión paralela, en la que la información obtenida en una sesión se utilice en otra diferente.

Si se viola alguna de estas reglas, es muy probable que se quebrante el protocolo. Aquí, se violaron las cuatro reglas con consecuencias desastrosas.

Ahora regresemos y demos un vistazo más de cerca a la figura 8-32. ¿Estamos seguros de que este protocolo no está sujeto a un ataque de reflexión? Quizás. Es muy sutil. Trudy fue capaz de derrotar nuestro protocolo utilizando un ataque de reflexión porque pudo abrir una segunda sesión con Bob y lo engañó para que contestara sus propias preguntas. ¿Qué habría pasado si Alice fuera una computadora de propósito general que también aceptara sesiones múltiples, en lugar de una persona en una computadora? Veamos lo que puede hacer Trudy.

Para saber cómo funciona el ataque de Trudy, vea la figura 8-35. Alice empieza por anunciar su identidad en el mensaje 1. Trudy intercepta ese mensaje y comienza su propia sesión con el mensaje 2, en donde afirma ser Bob. Nuevamente sombreamos los mensajes de la sesión 2. Alice responde al mensaje 2 con el mensaje 3 y dice: "¿Así que eres Bob? Demuéstralo". En este punto Trudy está atorada porque no puede probar que es Bob.

¿Qué hace Trudy ahora? Regresa a la primera sesión, en donde le toca enviar un desafío y envía el R_A que obtuvo en el mensaje 3. Alice responde amablemente a él en el mensaje 5, y de esta forma proporciona a Trudy la información que necesita para enviar el mensaje 6 en la sesión 2. En este punto, Trudy está prácticamente del otro lado porque ha respondido exitosamente al desafío de Alice en la sesión 2. Ahora puede cancelar la sesión 1, enviar cualquier número anterior durante el resto de la sesión 2 y tendrá una sesión autentificada con Alice en la sesión 2.

Pero Trudy es vil y realmente desea insistir. En lugar de enviar cualquier número anterior para completar la sesión 2, espera hasta que Alice envía el mensaje 7, el desafío de Alice para la sesión 1. Por supuesto que Trudy no sabe cómo responder, por lo que utiliza una vez más el ataque de reflexión y regresa R_{A2} como el mensaje 8. Alice encripta de manera apropiada R_{A2} en el mensaje 9. Trudy ahora cambia a la sesión 1 y envía a Alice el número que desea en el mensaje 10, copiado de manera conveniente a partir de lo que Alice envió en el mensaje 9. En este punto Trudy tiene dos sesiones completamente autenticadas con Alice.

Este ataque tiene un resultado ligeramente distinto al ataque del protocolo de tres mensajes que vimos en la figura 8-34. Esta vez, Trudy tiene dos conexiones autentificadas con Alice. En el ejemplo anterior, tenía una conexión autentificada con Bob. Aquí, si hubiéramos aplicado todas las reglas de protocolos de autentificación generales que analizamos antes, podríamos haber detenido este ataque. Para un análisis detallado de este tipo de ataques y cómo frustrarlos, consulte a Bird y colaboradores (1993). Éstos también nos muestran cómo es posible construir de manera sistemática protocolos que demuestren ser correctos. No obstante, el protocolo más simple de este tipo es algo complicado, por lo que ahora vamos a ver una clase diferente de protocolo que también funciona.

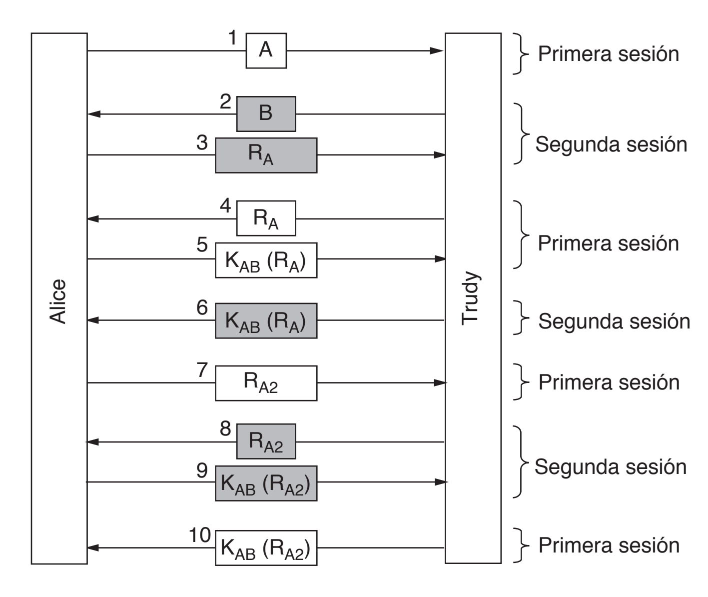


Figura 8-35. Un ataque de reflexión al protocolo de la figura 8-32.

En la figura 8-36 se muestra el nuevo protocolo de autentificación (Bird y colaboradores, 1993). Utiliza un HMAC del tipo que vimos cuando estudiamos IPsec. Para empezar, Alice envía a Bob el nonce R_A como el mensaje 1. Bob responde seleccionando su propio nonce, R_B, y lo regresa junto con un HMAC. Para formar el HMAC, se construye una estructura de datos que consiste en el nonce de Alice, el nonce de Bob, sus identidades y la clave secreta compartida, K_{AB}. Después, a esta estructura de datos se le aplica un hash en el HMAC; por ejemplo, mediante el uso de SHA-1. Cuando Alice recibe el mensaje 2, tiene un R_A (que ella misma eligió), un R_B que llega como texto plano, las dos identidades y la clave secreta K_{AB}, que ya conocía desde un principio, por lo que ella misma puede calcular el HMAC. Si corresponde con el HMAC del mensaje, Alice sabe que está hablando con Bob porque Trudy no conoce K_{AB} y, por lo tanto, no sabe cuál HMAC enviar. Alice le responde a Bob con un HMAC que contiene sólo los dos nonces.

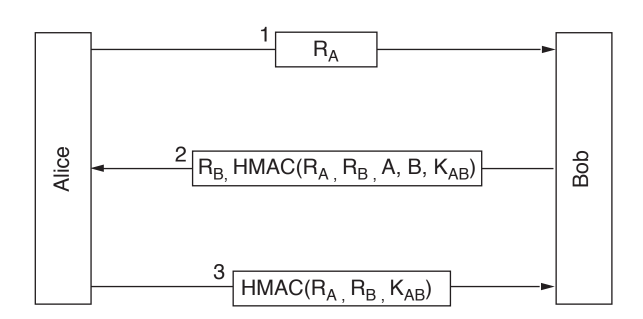


Figura 8-36. Autentificación mediante el uso de códigos HMAC.

¿Puede Trudy destruir de alguna forma este protocolo? No, porque ella no puede obligar a ninguna de las dos partes a encriptar o aplicar un hash a un valor que ella seleccione, como sucedió en los casos mostrados en las figuras 8-34 y 8-35. Ambos códigos HMAC incluyen valores elegidos por la parte emisora, algo que Trudy no puede controlar.

El uso de códigos HMAC no es la única forma de utilizar esta idea. Un esquema alternativo que se utiliza con mucha frecuencia en lugar de calcular el HMAC a través de una serie de elementos es el de encriptar dichos elementos de manera secuencial, mediante el uso del encadenamiento de bloques de sistema de cifrado.

8.7.2 Establecimiento de una clave compartida: el intercambio de claves de Diffie-Hellman

Hasta ahora hemos supuesto que Alice y Bob comparten una clave secreta. Suponga que no lo hacen (porque hasta el momento no hay una PKI aceptada de manera universal para firmar y distribuir certificados). ¿Cómo pueden establecer una? Una forma sería que Alice llamara a Bob y le diera su clave por teléfono, pero quizás él empezaría por decir: "¿Cómo sé que eres Alice y no Trudy?" Podrían tratar de establecer una reunión, en la que cada uno trajera un pasaporte, una licencia de conducir y tres tarjetas de crédito, pero por ser gente ocupada, tal vez no encuentren una fecha aceptable para los dos en meses. Por fortuna, tan increíble como pueda parecer, hay una forma para que personas totalmente extrañas establezcan una clave secreta compartida a la vista de todos, incluso si Trudy está grabando con cuidado cada mensaje.

El protocolo que permite que extraños establezcan una clave secreta compartida se conoce como **intercambio de claves de Diffie-Hellman** (Diffie y Hellman, 1976) y funciona de la siguiente manera. Alice y Bob tienen que estar de acuerdo en dos números grandes, n y g, en donde n sea un número primo, $(n - 1)/2$ también sea un número primo y se apliquen ciertas condiciones a g. Estos números pueden ser públicos, por lo que cualquiera de ellos (Alice o Bob) sólo puede elegir tanto n como g y decírselos al otro abiertamente. Ahora Alice elige un número grande (digamos, de 1024 bits), x, y lo mantiene en secreto. De manera similar, Bob elige un número secreto grande, y.

Para iniciar el protocolo de intercambio de claves, Alice envía a Bob un mensaje que contiene (n, g, $g^x \bmod n$), como se muestra en la figura 8-37. Para responder, Bob envía a Alice un mensaje que contiene $g^y \bmod n$. Ahora Alice eleva el número que Bob le envío a la x potencia módulo n para obtener $(g^y \bmod n)^x \bmod n$. Bob realiza una operación similar para obtener $(g^x \bmod n)^y \bmod n$. Por las leyes de la aritmética modular, ambos cálculos dan como resultado $g^{xy} \bmod n$. Y he aquí que, como por arte de magia, Alice y Bob de repente comparten una clave secreta, $g^{xy} \bmod n$.

Por supuesto que Trudy ha visto ambos mensajes. Ella conoce tanto g como n a partir del mensaje 1. Si pudiera calcular x y y, podría adivinar la clave secreta. El problema es que, con sólo $g^x \bmod n$ no le es

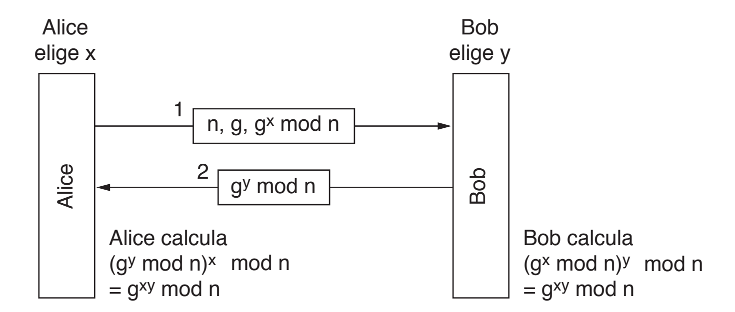


Figura 8-37. El intercambio de claves de Diffie-Hellman.

posible encontrar x. No se conoce ningún algoritmo práctico para calcular logaritmos discretos módulo con un número primo muy grande.

Para que el ejemplo sea más concreto, utilizaremos los valores (que no son para nada realistas) de $n = 47$ y $g = 3$. Alice elige $x = 8$ y Bob elige $y = 10$. Ambos valores se mantienen en secreto. El mensaje de Alice para Bob es (47, 3, 28) porque 3^8 mod 47 es 28. El mensaje de Bob para Alice es (17). Ella calcula 17^8 mod 47, lo cual es 4. Él calcula 28^{10} mod 47, lo cual es 4. Ahora ambos han determinado de manera independiente que la clave secreta es 4. Para encontrar la clave, Trudy tiene que resolver la ecuación 3^x mod $47 = 28$, lo cual se puede hacer mediante una búsqueda exhaustiva de números pequeños como éstos, pero no cuando todos los números tienen una longitud de cientos de bits. Todos los algoritmos conocidos en la actualidad tardan mucho, incluso en supercomputadoras con gran capacidad de procesamiento en paralelo y muy veloces.

A pesar de la elegancia del algoritmo de Diffie-Hellman, hay un problema: cuando Bob obtiene el triple (47, 3, 28), ¿cómo sabe que proviene de Alice y no de Trudy? No hay forma de que pueda saberlo. Por desgracia, Trudy puede explotar este hecho para engañar tanto a Alice como a Bob, como se ilustra en la figura 8-38. Aquí, mientras Alice y Bob están eligiendo x y y, respectivamente, Trudy elige su propio número aleatorio, z. Alice envía el mensaje 1 dirigido a Bob. Trudy lo intercepta y envía el mensaje 2 a Bob, en el que utiliza los valores g y n correctos (que de todas formas son públicos) pero con su propio valor z en lugar de x. También regresa el mensaje 3 a Alice. Más tarde Bob envía el mensaje 4 a Alice, el cual Trudy vuelve a interceptar y lo conserva.

Ahora todos realizan la aritmética modular. Alice calcula la clave secreta como g^{xz} mod n, al igual que Trudy (para los mensajes destinados a Alice). Bob calcula g^{yz} mod n al igual que Trudy (para los mensajes destinados a Bob). Alice cree que está hablando con Bob, por lo que establece una clave de sesión (con Trudy). Bob también hace lo mismo. Cada mensaje que Alice envía en la sesión encriptada, Trudy lo captura, almacena, modifica (si lo desea) y después lo pasa (lo cual es opcional) a Bob. De manera similar,

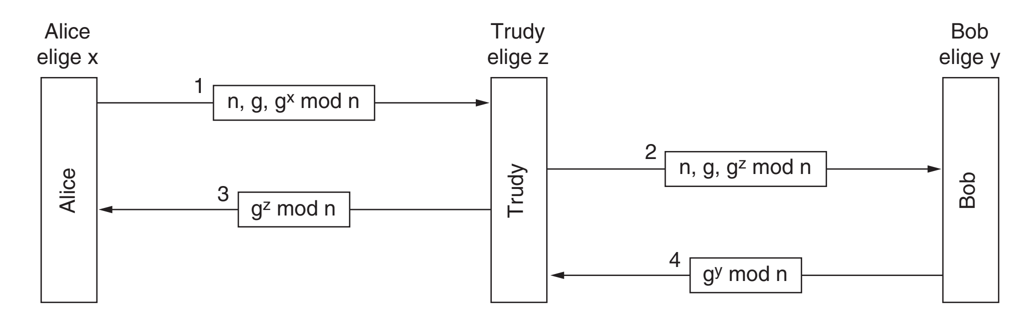


Figura 8-38. El ataque del intermediario (*man-in-the-middle*).

en la otra dirección, Trudy ve todo y puede modificar todos los mensajes a voluntad, mientras que Alice y Bob están con la creencia de que tienen un canal seguro. Por esta razón, a este ataque se le conoce como **ataque del intermediario** (*man-in-the-middle attack*). También se le conoce como **ataque de la brigada de cubeta** (*bucket brigade attack*), ya que se parece vagamente a un antiguo Departamento de Bomberos en el que éstos pasan cubetas en fila desde el camión cisterna hacia el incendio.

8.7.3 Autentificación mediante el uso de un centro de distribución de claves

Establecer un secreto compartido con un extraño casi funcionó, pero no del todo. Por otro lado, tal vez no valía la pena hacerlo en primer lugar (ataque de las uvas agrias). Para hablar de esta manera con *n* personas, serían necesarias *n* claves. Para las personas populares, la administración de claves podría volverse una verdadera carga, en especial si cada una tuviera que almacenarse por separado en una tarjeta de chip plástica.

Hay un método diferente: introducir un centro de distribución de claves de confianza. En este modelo, cada usuario tiene una clave compartida con el KDC, que se encarga de administrar las claves de sesión y autentificación. En la figura 8-39 se muestran el protocolo de autentificación KDC más simple conocido, que involucra a dos partes, y un KDC de confianza.

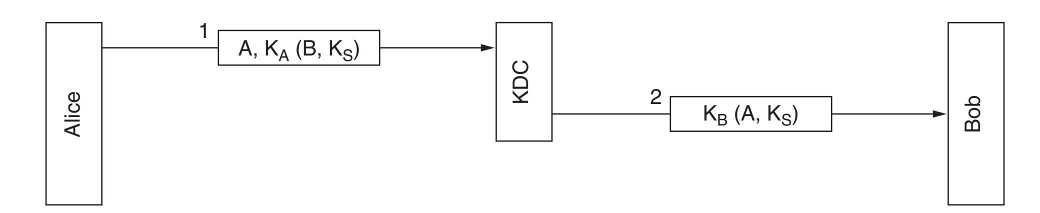


Figura 8-39. Un primer intento con un protocolo de autentificación mediante el uso de KDC.

La idea detrás de este protocolo es simple: Alice elige una clave de sesión K_S, e indica al KDC que desea hablar con Bob mediante el uso de K_S. Este mensaje se encripta con la clave secreta que Alice comparte (sólo) con el KDC: K_A. El KDC desencripta este mensaje; después extrae la identidad de Bob y la clave de sesión. A continuación construye un nuevo mensaje que contiene la identidad de Alice y la clave de sesión, y envía este mensaje a Bob. Esta encriptación se realiza con K_B, la clave secreta que Bob comparte con el KDC. Cuando Bob desencripta el mensaje, se entera de que Alice desea hablar con él y cuál clave desea utilizar.

Aquí, la autentificación se lleva a cabo sin problemas. El KDC sabe que el mensaje 1 debe provenir de Alice, puesto que nadie más podría encriptarlo con la clave secreta de Alice. De manera similar, Bob sabe que el mensaje 2 debe provenir del KDC, en quien él confía, puesto que nadie más sabe su clave secreta.

Por desgracia, este protocolo tiene un defecto grave. Trudy necesita algo de dinero, por lo que idea algún servicio legítimo que puede realizar para Alice, realiza una oferta atractiva y obtiene el trabajo. Después de realizar el trabajo, Trudy solicita amablemente a Alice que le pague por transferencia bancaria. A continuación Alice establece una clave de sesión con su banquero, Bob. Después envía a Bob un mensaje en el que le pide que transfiera dinero a la cuenta de Trudy.

Mientras tanto, Trudy regresa a sus antiguos hábitos: espiar la Red. Copia el mensaje 2 de la figura 8-39 junto con la solicitud de transferencia de dinero que le sigue. Más tarde, lo repite para Bob, quien piensa: “Alice debe haber contratado nuevamente a Trudy. Ésta seguramente hace un magnífico trabajo”. Después Bob transfiere una cantidad igual de dinero de la cuenta de Alice a la de Trudy. Algún tiempo después del quincuagésimo par de mensajes, Bob sale corriendo de la oficina con el fin de encontrar a

Trudy para ofrecerle un gran préstamo de manera que pueda ampliar su, obviamente, exitoso negocio. Este problema se conoce como el **ataque de repetición**.

Hay varias soluciones posibles para el ataque de repetición. La primera es incluir una estampa de tiempo en cada mensaje. Así, si alguien recibe un mensaje obsoleto, lo puede descartar. El problema con este método es que los relojes en una red nunca están sincronizados de manera exacta, por lo que debe haber un intervalo durante el cual sea válida la estampa de tiempo. Trudy puede repetir el mensaje durante este intervalo y salirse con la suya.

La segunda solución es colocar un nonce en cada mensaje. Así, cada parte tiene que recordar todos los nonces anteriores y rechazar cualquier mensaje que contenga un nonce que se haya utilizado antes. Pero los nonces se deben recordar por siempre, con el fin de que Trudy no trate de repetir un mensaje con cinco años de antigüedad. Además, si una máquina falla y pierde su lista de nonces, nuevamente es vulnerable a un ataque de repetición. Las estampas de tiempo y los nonces se pueden combinar para limitar cuánto tiempo deben recordarse los nonces, pero es evidente que el protocolo se va a complicar mucho más.

Un método mucho más sofisticado para la autentificación mutua es el de utilizar un protocolo de desafío-respuesta de múltiples vías. Un ejemplo bien conocido de dicho protocolo es el de **autentificación de Needham-Schroeder** (Needham y Schroeder, 1978); en la figura 8-40 se muestra una variante de este protocolo.

El protocolo comienza así: Alice indica al KDC que desea hablar con Bob. Este mensaje contiene como nonce un número aleatorio grande, R_A. El KDC regresa el mensaje 2 que contiene el número aleatorio de Alice, una clave de sesión y un boleto que ella puede enviar a Bob. El objetivo del número aleatorio R_A es asegurar a Alice que el mensaje 2 está actualizado y que no es una repetición. También se incluye la identidad de Bob, en caso de que Trudy tenga algunas ideas graciosas acerca de reemplazar B en el mensaje 1 con su propia identidad, con el fin de que el KDC encripte el boleto al final del mensaje 2 con K_T en lugar de K_B. El boleto encriptado con K_B se incluye dentro del mensaje encriptado para evitar que Trudy lo reemplace con algo más cuando el mensaje regrese hacia Alice.

Ahora Alice envía el boleto a Bob, junto con un nuevo número aleatorio R_{A2}, encriptado con la clave de sesión, K_S. En el mensaje 4, Bob regresa K_S $(R_{A2} - 1)$ para demostrar a Alice que está hablando con el verdadero Bob. No hubiera funcionado el hecho de regresar K_S (R_{A2}), puesto que Trudy podría haberlo robado del mensaje 3.

Después de recibir el mensaje 4, Alice está convencida de que está hablando con Bob y de que hasta el momento no es posible que se hayan utilizado repeticiones. Después de todo, generó R_{A2} hace tan sólo unos cuantos milisegundos. El propósito del mensaje 5 es convencer a Bob de que sí es Alice con la que está hablando, y de que aquí tampoco se han utilizado repeticiones. Al hacer que cada una de las dos partes genere un desafío y una respuesta, se elimina cualquier posibilidad de algún tipo de ataque de repetición.

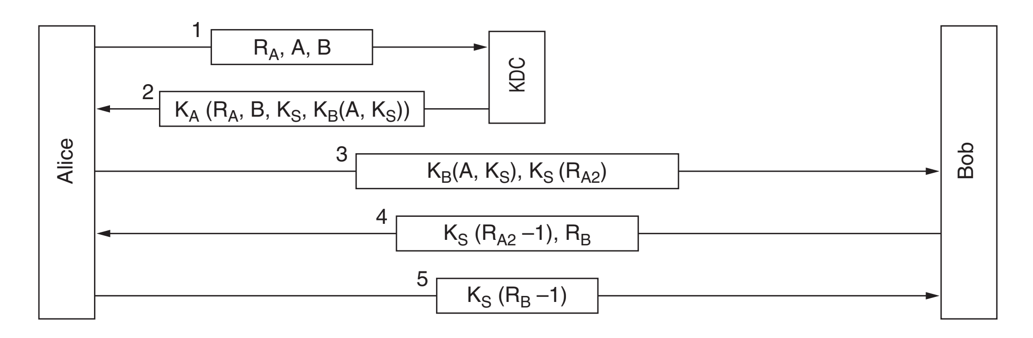


Figura 8-40. El protocolo de autentificación de Needham-Schroeder.

Aunque este protocolo parece muy sólido, tiene una ligera debilidad. Si Trudy se las arregla para conseguir una sesión de clave antigua en texto plano, puede iniciar una nueva sesión con Bob si repite el mensaje 3 que corresponde a la clave comprometida y lo convence de que ella es Alice (Denning y Sacco, 1981). Esta vez puede saquear la cuenta bancaria de Alice sin tener que realizar ni siquiera una vez el servicio legítimo.

Más tarde, Needham y Schroeder (1987) publicaron un protocolo que corrige este problema. En el mismo número de la misma publicación, Otway y Rees (1987) también dieron a conocer un protocolo que resuelve el problema en una forma más corta. La figura 8-41 muestra un protocolo Otway-Rees con unas ligeras modificaciones.

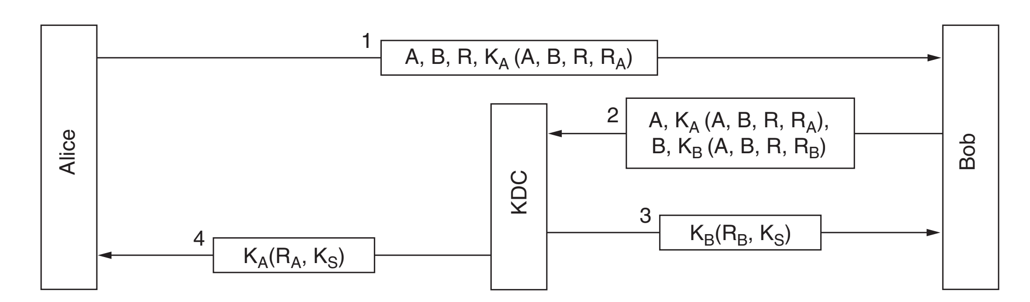


Figura 8-41. El protocolo de autentificación de Otway-Rees (con una ligera simplificación).

En el protocolo de Otway-Rees, al inicio Alice genera un par de números aleatorios: R, que se utilizará como identificador común, y R_A, que Alice utilizará para desafiar a Bob. Cuando Bob recibe este mensaje, construye uno nuevo a partir de la parte encriptada del mensaje de Alice y de su propio mensaje análogo. Ambas partes encriptadas con K_A y K_B identifican a Alice y a Bob, contienen el identificador común y también contienen un desafío.

El KDC verifica si el R en ambas partes es el mismo. Podría ser distinto si Trudy falsificó R en el mensaje 1 o reemplazó parte del mensaje 2. Si los dos números R coinciden, el KDC considera que el mensaje de solicitud de Bob es válido. Después genera una clave de sesión y la encripta dos veces, una para Alice y la otra para Bob. Cada mensaje contiene el número aleatorio del receptor, como prueba de que el KDC, y no Trudy, generó el mensaje. En este punto tanto Alice como Bob poseen la misma clave de sesión y pueden empezar a comunicarse. La primera vez que intercambian mensajes de datos, cada uno puede ver que el otro tiene una copia idéntica de K_S, por lo que entonces la autentificación está completa.

8.7.4 Autentificación mediante el uso de Kerberos

Kerberos es un protocolo de autentificación que se utiliza en muchos sistemas reales (incluyendo Windows 2000 y versiones posteriores); está basado en una variante de Needham-Schroeder. Su nombre proviene del perro de múltiples cabezas de la mitología griega que custodiaba la entrada a Hades (se supone que para mantener fuera a las personas indeseables). Kerberos se diseñó en el MIT para permitir que los usuarios de estaciones de trabajo accedieran a los recursos de red de una forma segura. Su mayor diferencia respecto a Needham-Schroeder es la suposición de que todos los relojes están bien sincronizados. El protocolo ha pasado por varias iteraciones. V5 es la versión más utilizada en la industria y se define en el RFC 4120. La versión anterior, V4, se retiró después de haber encontrado varias fallas graves (Yu y colaboradores, 2004). V5 presenta una mejora con respecto a V4 con muchos cambios pequeños al protocolo y ciertas características mejoradas, como el hecho de que ya no depende del DES, que ahora es obsoleto. Para mayor información, consulte a Neumam y Ts'o (1994).

Kerberos involucra a tres servidores además de Alice (una estación de trabajo cliente):

1. Servidor de autentificación (AS): verifica usuarios durante el inicio de sesión.
2. Servidor de otorgamiento de *tickets* (TGS): emite la “prueba de tickets de identidad”.
3. Bob, el servidor: hace el trabajo que Alice desea.

El AS es similar a KDC en cuanto a que comparte una contraseña secreta con cada usuario. El trabajo del TGS es emitir “tickets” que puedan convencer a los servidores reales de que el portador de un ticket TGS es realmente quien afirma ser.

Para iniciar una sesión, Alice se sienta frente a una estación de trabajo pública arbitraria e introduce su nombre. La estación de trabajo envía su nombre y el nombre del TGS al AS en texto plano, como se muestra en el mensaje 1 de la figura 8-42. Lo que regresa es una clave de sesión y un ticket, K_{TGS} (A, K_S, *t*), destinado para el TGS. La clave de sesión se encripta mediante la clave secreta de Alice, de forma que sólo ella pueda desencriptarla. Sólo hasta que llega el mensaje 2 es cuando la estación de trabajo pide la contraseña —ni un segundo antes—. A continuación, dicha contraseña se utiliza para generar K_A con el fin de desencriptar el mensaje 2 y obtener la clave de sesión.

En este punto, la estación de trabajo sobrescribe la contraseña de Alice para asegurarse de que sólo esté dentro de la estación de trabajo durante unos cuantos milisegundos, a lo más. Si Trudy intenta iniciar sesión como Alice, la contraseña que teclee será errónea y la estación de trabajo detectará esto porque la parte estándar del mensaje 2 será incorrecta.

Después de iniciar sesión, Alice podría decirle a la estación de trabajo que desea contactar a Bob, el servidor de archivos. A continuación, la estación de trabajo envía el mensaje 3 al TGS y le pide un ticket para utilizarlo con Bob. El elemento clave en esta petición es el ticket K_{TGS} (A, K_S, t), que se encripta con la clave secreta del TGS y que se utiliza como prueba de que el emisor es realmente Alice. El TGS responde en el mensaje 4 con la creación de una clave de sesión K_{AB}, para que Alice la utilice con Bob. Se regresan dos versiones de dicha clave. La primera se encripta sólo con K_S, de manera que Alice pueda leerla. La segunda es otro ticket que se encripta con la clave de Bob, K_B, de manera que él pueda leerla.

Trudy puede copiar el mensaje 3 y tratar de utilizarlo nuevamente, pero se frustrarán sus planes debido a la estampa de tiempo encriptada, *t*, la cual se envía junto con ese mensaje. Trudy no puede reemplazar la estampa de tiempo con una más reciente puesto que no conoce K_S, la clave de sesión que Alice utiliza para hablar con el TGS. Incluso si Trudy repite con rapidez el mensaje 3, todo lo que obtendrá será otra

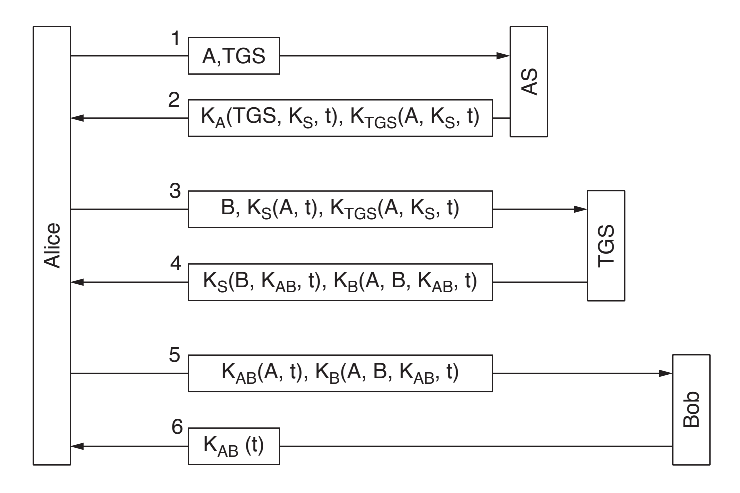


Figura 8-42. La operación de Kerberos V5.

copia del mensaje 4, el cual no pudo desencriptar la primera vez y que tampoco podrá desencriptar esta segunda vez.

Ahora Alice puede enviar K_{AB} a Bob por medio del nuevo ticket, para establecer una sesión con él (mensaje 5). Este intercambio también tiene una estampa de tiempo. La respuesta opcional (mensaje 6) es la prueba para Alice de que realmente está hablando con Bob, no con Trudy.

Después de esta serie de intercambios, Alice se puede comunicar con Bob bajo la cobertura de K_{AB}. Si más adelante necesita hablar con otro servidor, como Carol, simplemente repite el mensaje 3 al TGS, sólo que ahora especifica *C* en lugar de *B*. El TGS responderá rápidamente con un boleto encriptado con K_C que Alice pueda enviar a Carol, y que ésta aceptará como prueba de que proviene de Alice.

El objetivo de todo este trabajo es que ahora Alice puede acceder a servidores a través de toda la Red de forma segura, y que su contraseña no tiene que pasar a través de la Red. De hecho, sólo tuvo que estar en su propia estación de trabajo por unos cuantos milisegundos. Sin embargo, cabe mencionar que cada servidor realiza su propia autorización. Cuando Alice presenta su ticket a Bob, esto simplemente le prueba quién lo envió. Él es quien determina las acciones que Alice tiene permitido realizar.

Puesto que los diseñadores de Kerberos no esperaron que todo el mundo confiara en un solo servidor de autentificación, establecieron reglas para tener múltiples **dominios** (*realms*), cada uno con su propio AS y TGS. Para obtener un ticket para un servidor de un dominio distante, Alice podría solicitar a su propio TGS un ticket aceptado por el TGS en el dominio distante. Si el TGS distante se ha registrado con el TGS local (de la misma manera en que lo hacen los servidores locales), este último le dará a Alice un ticket válido en el TGS distante. Así ella puede hacer negocios allá, como obtener tickets para servidores de ese dominio. Sin embargo, hay que tener en cuenta que para que las partes de dos dominios hagan negocios, cada una debe confiar en el TGS de la otra. De lo contrario, no podrán hacer negocios.

8.7.5 Autentificación mediante el uso de criptografía de clave pública

La autentificación mutua también se puede realizar mediante la criptografía de clave pública. Para empezar, Alice necesita obtener la clave pública de Bob. Si existe una PKI con un servidor de directorios que proporciona certificados para claves públicas, Alice puede pedir la de Bob, que en la figura 8-43 se muestra como el mensaje 1. La respuesta, que va en el mensaje 2, es un certificado X.509 que contiene la clave pública de Bob. Cuando Alice verifica que la firma es correcta, envía a Bob un mensaje que contiene su identidad y un nonce.

Cuando Bob recibe este mensaje, no tiene idea de si proviene de Alice o de Trudy, pero sigue adelante y pide al servidor de directorio la clave pública de Alice (mensaje 4) la cual obtiene de inmediato

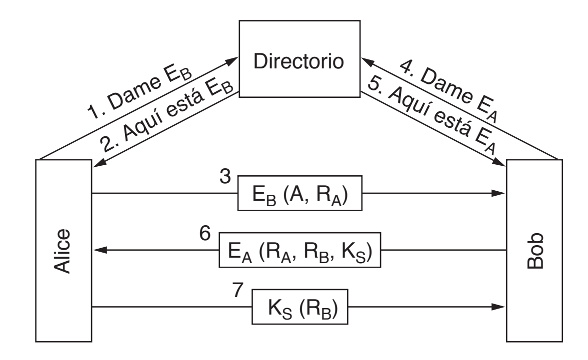


Figura 8-43. Autentificación mutua mediante el uso de la criptografía de clave pública.

(mensaje 5). A continuación envía a Alice el mensaje 6, el cual contiene el R_A de Alice, su propio nonce, R_B y una clave de sesión propuesta, K_S.

Cuando Alice recibe el mensaje 6, lo desencripta mediante su clave privada. Ve el R_A en dicho mensaje, lo cual le trae alivio. El mensaje debe provenir de Bob, puesto que Trudy no tiene forma de determinar R_A. Además, debe ser una actualización y no una repetición, puesto que acaba de enviarle a Bob el R_A. Alice accede a la sesión, para lo cual envía el mensaje 7. Cuando Bob ve R_B encriptado con la clave de sesión que acaba de generar, sabe que Alice obtuvo el mensaje 6 y verificó R_A. Ahora Bob es un campista feliz.

¿Qué puede hacer Trudy para tratar de destruir este protocolo? Puede fabricar el mensaje 3 y engañar a Bob para que investigue a Alice, pero ésta verá un R_A que ella no envió y ya no proseguirá. Trudy no podrá falsificar el mensaje 7 para Bob porque no conoce R_B o K_S y no puede determinarlos sin la clave privada de Alice. Se acabó su buena suerte.

8.8 SEGURIDAD DE CORREO ELECTRÓNICO

Cuando se envía un mensaje de correo electrónico entre dos sitios distantes, por lo general pasará por docenas de máquinas en el trayecto. Cualquiera de ellas puede leer y registrar el mensaje para usarlo después. En la práctica no existe la privacidad, a pesar de lo que mucha gente piensa. Sin embargo, muchas personas desearían tener la capacidad de enviar correo electrónico que sólo pueda ser leído por el destinatario y por nadie más: ni siquiera su jefe ni su gobierno. Este deseo ha estimulado a varias personas y grupos para aplicar al correo electrónico los principios criptográficos que estudiamos antes para producir correo electrónico seguro. En las siguientes secciones analizaremos un sistema de correo electrónico seguro muy utilizado, PGP, y después mencionaremos otro más: S/MIME. Para información adicional acerca del correo electrónico seguro, consulte a Kaufman y colaboradores (2002), y también a Schneier (1995).

8.8.1 PGP: Privacidad Bastante Buena

Nuestro primer ejemplo, **PGP** (**Privacidad Bastante Buena**, del inglés *Pretty Good Privacy*), es en esencia la idea original de una persona, Phil Zimmermann (1995a, 1995b). Zimmermann es un defensor de la privacidad cuyo lema es: “Si la privacidad se prohíbe, solamente los delincuentes tendrán privacidad”. Liberado en 1991, PGP es un paquete de seguridad de correo electrónico completo que proporciona privacidad, autentificación, firmas digitales y compresión, todo en una forma fácil de utilizar. Además, todo el paquete (incluyendo el código fuente) se distribuye sin costo alguno a través de Internet. Debido a su cualidad, precio (cero) y fácil disponibilidad en las plataformas UNIX, Linux, Windows y Mac OS, se utiliza ampliamente hoy en día.

Para encriptar los datos, PGP utiliza un sistema de cifrado de bloque llamado **IDEA** (**Algoritmo Internacional de Encriptación de Datos**, del inglés *Internacional Data Encryption Algorithm*), el cual utiliza claves de 128 bits. Se diseñó en Suiza en la época en que el DES se veía como defectuoso y el AES todavía no se inventaba. En concepto, IDEA es similar a DES y AES: mezcla los bits en una serie de rondas, pero los detalles de las funciones de mezcla son diferentes de los de DES y AES. La administración de claves utiliza RSA y la integridad de datos utiliza MD5, temas que ya hemos analizado.

PGP también se ha visto envuelto en controversias desde su primer día (Levy, 1993). Debido a que Zimmermann no hizo nada para evitar que otras personas colocaran a PGP en Internet, en donde personas de todo el mundo tuvieran acceso a éste, el gobierno de Estados Unidos declaró que Zimmermann había violado las leyes de ese país, las cuales prohibían exportar equipo de guerra. La investigación del gobierno de Estados Unidos en contra de Zimmermann duró cinco años, pero con el tiempo se retiró, probablemente por dos razones. En primer lugar, no fue Zimmermann quien colocó a PGP en Internet,

por lo que su abogado alegó que *él* nunca exportó nada (y ahí entra la controversia de si la creación de un sitio web constituye o no una exportación). En segundo lugar, con el tiempo el gobierno se dio cuenta de que ganar el juicio significaba convencer al jurado de que un sitio web que contenía un programa de privacidad descargable estaba cubierto por la ley de tráfico de armas que prohíbe la exportación de material de guerra, como tanques, submarinos, aviones militares y armas nucleares. También es probable que los años de publicidad negativa no hayan sido de mucha ayuda.

Como observación adicional, las leyes de exportación son extrañas, por no decir algo peor. El gobierno consideró la colocación de código en un sitio web como una exportación ilegal y persiguió a Zimmermann durante cinco años por eso. Por otro lado, cuando alguien publicó todo el código fuente de PGP, que estaba en lenguaje C, en un libro (en una fuente grande con una suma de verificación en cada página para facilitar su digitalización) y después exportó dicho libro, no hubo problema con el gobierno debido a que los libros no se clasifican como equipo de guerra. La espada es más poderosa que la pluma, por lo menos para el Tío Sam.

Otro problema en el que se vio involucrado PGP tuvo que ver con la infracción de las patentes. La empresa que poseía la patente RSA, de nombre RSA Security, Inc., alegó que el uso que PGP hacía del algoritmo RSA infringía su patente, pero ese problema quedó resuelto a partir de la versión 2.6. Además, PGP utiliza otro algoritmo de encriptación patentado, IDEA, cuyo uso causó algunos problemas al principio.

Puesto que el código de PGP es abierto, diversas personas y grupos lo han modificado para producir varias versiones. Algunas de ellas se diseñaron para resolver los problemas con las leyes de equipo de guerra, otras se enfocaron en evitar el uso de algoritmos patentados y otras más deseaban convertirlo en un producto comercial de código fuente cerrado. Aunque en la actualidad las leyes sobre equipo de guerra se han liberalizado un poco (de lo contrario los productos que utilizan AES no podrían exportarse de Estados Unidos) y la patente de RSA caducó en septiembre de 2000, el legado de todos estos problemas es que hay en circulación varias versiones incompatibles de PGP, bajo diversos nombres. El análisis que veremos a continuación se enfoca en el PGP clásico, que es la versión más antigua y simple. En el RFC 2440 se describe otra versión popular, Open PGP. Otra que también es popular se conoce como GNU Privacy Guard.

PGP utiliza de manera intencional algoritmos criptográficos existentes en lugar de inventar nuevos. Se basa en su mayor parte en algoritmos que han soportado revisiones extensas de iguales y que no fueron diseñados o influenciados por ninguna agencia gubernamental para tratar de debilitarlos. Para las personas que desconfían del gobierno, esta propiedad es una gran ventaja.

PGP soporta la compresión de texto, la confidencialidad y firmas digitales, además de que proporciona características de administración extensa de claves, pero por extraño que parezca, no proporciona características de correo electrónico. Es como un preprocesador que recibe texto plano como entrada y produce texto cifrado firmado en base64 como salida. Desde luego que esta salida se puede enviar por correo electrónico después. Algunas implementaciones de PGP llaman a un agente de usuario como paso final para enviar realmente el mensaje.

Para ver cómo funciona PGP, consideremos el ejemplo que se muestra en la figura 8-44. Aquí, Alice desea enviar de forma segura un mensaje en texto plano firmado, P, a Bob. Tanto Alice como Bob tienen claves RSA privadas (D_X) y públicas (E_X). Vamos a suponer que cada uno conoce la clave pública del otro; analizaremos la administración de claves PGP dentro de poco.

Para empezar, Alice invoca el programa PGP en su computadora. Lo primero que hace PGP es aplicar un hash al mensaje de Alice, P, mediante MD5, y después encripta el hash resultante mediante su clave RSA privada, D_A. Cuando Bob obtiene el mensaje, puede desencriptar el hash con la clave pública de Alice y verificar que dicho hash sea correcto. Incluso si alguien más (por ejemplo, Trudy) pudiera adquirir el hash en esta etapa y desencriptarlo con la clave pública conocida de Alice, la fuerza de MD5 garantiza que desde un punto de vista computacional sea imposible producir otro mensaje con el mismo hash MD5.

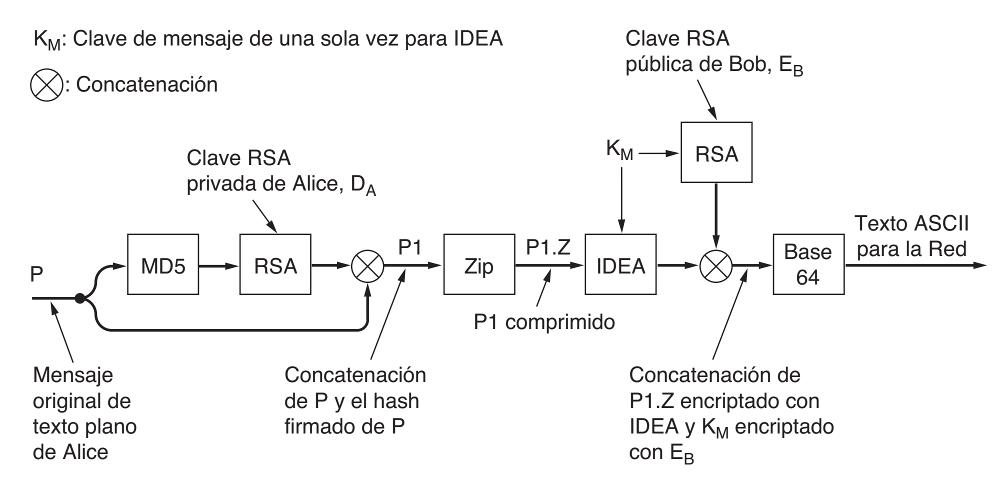


Figura 8-44. PGP en operación para enviar un mensaje.

El hash encriptado y el mensaje original están ahora concatenados en un solo mensaje, *P1*, y se comprimen mediante un programa ZIP, que utiliza el algoritmo Ziv-Lempel (Ziv y Lempel, 1977). Llamemos *P1.Z* al resultado obtenido de este paso.

A continuación, PGP pide a Alice una entrada aleatoria. Se utilizan tanto el contenido como la velocidad de tecleo para generar una clave de mensaje IDEA de 128 bits, K_M (se conoce como clave de sesión en la literatura de PGP, pero en realidad éste es un nombre inapropiado puesto que no hay una sesión). Ahora, K_M se utiliza para encriptar *P1.Z* con IDEA en modo de retroalimentación de sistema de cifrado. Además, K_M se encripta con la clave pública de Bob, E_B. Después, estos dos componentes se concatenan y convierten a base64, como vimos en la sección sobre MIME en el capítulo 7. El mensaje resultante contiene sólo letras, dígitos y los símbolos +, / y =, lo que significa que se puede colocar en el cuerpo de un RFC 822 y se puede esperar que llegue sin modificaciones.

Cuando Bob recibe el mensaje, invierte la codificación base64 y desencripta la clave IDEA mediante su clave RSA privada. Bob puede utilizar esta clave para desencriptar el mensaje con el fin de obtener *P1.Z*. Después de descomprimirlo, separa el texto plano del hash encriptado y lo desencripta mediante la clave pública de Alice. Si el hash del texto plano coincide con su propio cálculo de MD5, Bob sabe que *P* es el mensaje correcto y que proviene de Alice.

Vale la pena señalar que aquí RSA sólo se utiliza en dos lugares: para encriptar el hash MD5 de 128 bits y para encriptar la clave IDEA de 128 bits. Aunque RSA es lento, sólo tiene que encriptar 256 bits, no un gran volumen de datos. Además, los 256 bits de texto plano son demasiado aleatorios, por lo que se necesitará que Trudy realice una cantidad considerable de trabajo para determinar si una clave adivinada es correcta. IDEA realiza el trabajo rudo de encriptación, el cual es varias veces más rápido que RSA. Por lo tanto, PGP proporciona seguridad, compresión y una firma digital, y lo hace en una forma mucho más eficiente que el esquema ilustrado en la figura 8-19.

PGP soporta cuatro longitudes de clave RSA. Es responsabilidad del usuario seleccionar la más adecuada. Las longitudes son:

1. Casual (384 bits): en la actualidad se puede quebrantar con facilidad.
2. Comercial (512 bits): organizaciones de tres letras pueden quebrantarla.
3. Militar (1024 bits): nadie de este mundo puede quebrantarla.
4. Extraterrestre (2048 bits): ni siquiera alguien de otro planeta puede quebrantarla.

Puesto que RSA sólo se utiliza para dos pequeños cálculos, todo el mundo debería utilizar todo el tiempo las claves de categoría extraterrestre.

En la figura 8-45 se muestra el formato de un mensaje PGP clásico. También se utilizan muchos otros formatos. El mensaje tiene tres partes, las cuales contienen la clave IDEA, la firma y el mensaje, respectivamente. La parte de la clave no sólo contiene a ésta, sino también un identificador de clave, puesto que se permite que los usuarios tengan múltiples claves públicas.

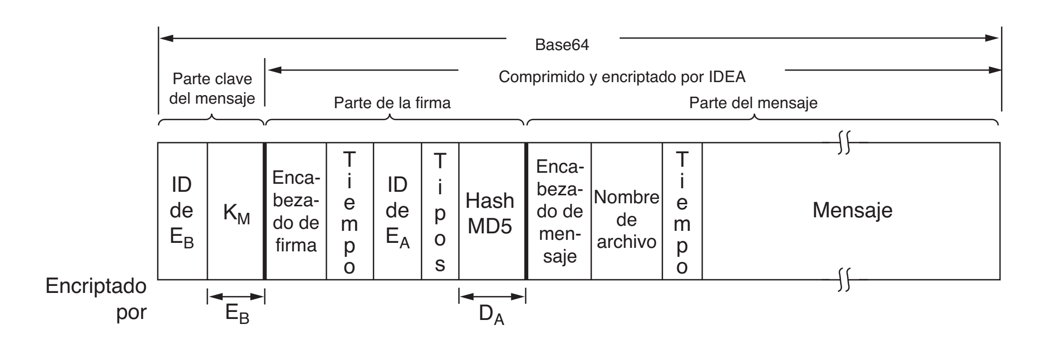


Figura 8-45. Un mensaje PGP.

La parte de firma contiene un encabezado, el cual no trataremos aquí. El encabezado va seguido por una estampa de tiempo, el identificador para la clave pública del emisor que se puede utilizar para desencriptar el hash de firma, cierta información de tipos que identifica los algoritmos utilizados (para permitir que MD6 y RSA2 se puedan utilizar cuando se inventen) y el hash encriptado mismo.

La parte del mensaje también contiene un encabezado, el nombre predeterminado del archivo que se va a utilizar si el receptor escribe el archivo en disco, una estampa de tiempo de creación de mensaje y, finalmente, el mensaje mismo.

La administración de claves ha recibido una gran cantidad de atención en PGP, puesto que es “el talón de Aquiles” de los sistemas de seguridad. La administración de claves funciona de la siguiente manera. Cada usuario mantiene dos estructuras de datos en forma local: un anillo de clave privada y un anillo de clave pública. El **anillo de clave privada** contiene uno o más pares de clave pública/privada personales. La razón para soportar múltiples pares por usuario es permitir que los usuarios cambien sus claves públicas con regularidad o cuando piensen que han sido comprometidas, sin invalidar los mensajes que están actualmente en preparación o en tránsito. Cada par tiene un identificador asociado, por lo que un emisor de mensajes puede indicar al destinatario cuál clave pública se utilizó para encriptarlo. Los identificadores de mensajes consisten en los 64 bits de menor orden de la clave pública. Los usuarios son los responsables de evitar conflictos en sus identificadores de clave pública. Las claves privadas en disco se encriptan mediante una contraseña especial (de tamaño arbitrario) para protegerlas contra ataques de robo.

El **anillo de clave pública** contiene las claves públicas de los contactos del usuario. Éstas se necesitan para encriptar las claves de mensaje asociadas con cada mensaje. Cada entrada en el anillo de clave pública no sólo contiene dicha clave pública, sino también su identificador de 64 bits y una indicación de qué tanto confía el usuario en la clave.

El problema que tratamos de solucionar aquí es el siguiente. Suponga que las claves públicas se mantienen en boletines electrónicos. Una forma en que Trudy puede leer el correo electrónico secreto de Bob es atacar el boletín electrónico y reemplazar la clave pública de Bob con una de su elección. Cuando más tarde Alice obtiene la clave que supuestamente pertenece a Bob, Trudy puede embestir a Bob mediante un ataque de brigada de cubeta.

Para evitar tales ataques, o por lo menos minimizar sus consecuencias, Alice necesita saber cuánto debe confiar en el elemento llamado "clave de Bob" de su anillo de clave pública. Si Alice sabe que Bob le proporcionó personalmente un CD-ROM que contiene la clave, puede establecer el valor de confianza en el nivel más alto. Este enfoque controlado por el usuario y descentralizado hacia la administración de claves públicas es el que marca la diferencia entre PGP y los esquemas centralizados PKI.

Sin embargo, algunas veces las personas obtienen claves públicas al consultar a un servidor de claves de confianza. Por esta razón, después de que se estandarizó el X.509, PGP soportó estos certificados al igual que el mecanismo de anillo de clave pública de PGP. Todas las versiones actuales de PGP ofrecen soporte para X.509.

8.8.2 S/MIME

La incursión de la IETF en lo que se refiere a la seguridad de correo electrónico, conocida como **S/MIME** (**MIME Seguro**), se describe en los documentos RFC 2632 al 2643. Ofrece autentificación, integridad de datos, confidencialidad y no repudio. También es muy flexible, pues soporta una variedad de algoritmos criptográficos. No es sorprendente, dado el nombre, que S/MIME se integre bien con MIME, lo cual le permite proteger todo tipo de mensajes. Por ejemplo, se define una gran variedad de nuevos encabezados MIME para contener firmas digitales.

S/MIME no tiene una jerarquía de certificados rígida que comienza en una sola raíz, lo cual había sido uno de los problemas políticos que condenaron al fracaso a un sistema anterior conocido como PEM (Correo con Privacidad Mejorada). En su lugar, los usuarios pueden tener varias anclas de confianza. Un certificado se considera válido siempre y cuando se pueda rastrear hacia alguna ancla en la que el usuario confíe. S/MIME utiliza los algoritmos y protocolos estándar que hemos examinado hasta ahora, por lo que no los analizaremos más aquí. Para más detalles, por favor consulte los RFC.

8.9 SEGURIDAD EN WEB

Hemos estudiado dos áreas importantes en donde es necesaria la seguridad: las comunicaciones y el correo electrónico. Podemos considerarlos como la sopa y el entremés. Ahora es tiempo del platillo principal: la seguridad en la web. La web es el lugar donde la mayoría de las Trudy vagan en la actualidad y tratan de realizar su trabajo sucio. En las siguientes secciones analizaremos algunos de los problemas y cuestiones que se relacionan con la seguridad en web.

La seguridad en web se puede dividir en tres partes. En primer lugar, ¿cómo se nombran los objetos y los recursos en forma segura? En segundo lugar, ¿cómo se pueden establecer conexiones seguras y autentificadas? En tercer lugar, ¿qué sucede cuando un sitio web envía a un cliente una pieza de código ejecutable? Después de ver algunas amenazas, examinaremos todas estas cuestiones.

8.9.1 Amenazas

Podemos leer en los periódicos los problemas de seguridad en los sitios web casi todas las semanas. La situación es verdaderamente muy sombría. Veamos algunos ejemplos de lo que ha sucedido en realidad. Primero, las páginas de inicio de numerosas organizaciones han sido atacadas y reemplazadas por nuevas páginas de inicio elegidas por los crackers (la prensa popular llama "hackers" a las personas que irrumpen en las computadoras, pero muchos programadores reservan ese término para los grandes programadores. Nosotros preferimos llamar "crackers" a las personas que irrumpen sin permiso a la computadora de otra persona). Entre los sitios que han sido atacados por los crackers se incluyen Yahoo!, el Ejército de Esta-

dos Unidos, la CIA, la NASA y el *New York Times*. En la mayoría de los casos, los crackers simplemente colocaron un texto gracioso y los sitios se repararon en unas cuantas horas.

Ahora veamos algunos casos más serios. Muchos sitios han sido derribados por ataques de negación de servicio, en los que el cracker inunda con tráfico el sitio, dejándolo incapaz de responder a las consultas legítimas. Con frecuencia, el ataque se realiza desde una gran cantidad de máquinas en las que el cracker ya ha irrumpido (ataques DDoS). Estos ataques son tan comunes que ya no son noticia, pero pueden costar al sitio atacado miles de dólares en pérdidas relacionadas con sus actividades comerciales.

En 1999, un cracker sueco irrumpió en el sitio web Hotmail de Microsoft y creó un sitio espejo que permitió que cualquier persona tecleara el nombre de un usuario de Hotmail y leyera todo su correo electrónico.

En otro caso, un cracker ruso de 19 años llamado Maxim irrumpió en un sitio web de comercio electrónico y robó 300 000 números de tarjetas de crédito. Después contactó a los dueños del sitio y les dijo que, si no le pagaban $100 000, publicaría en Internet todos los números de las tarjetas. Los dueños no cedieron a su chantaje y Maxim publicó dichos números, con lo cual dañó a muchas víctimas inocentes.

Otro caso fue el de un estudiante de California de 23 años que envió por correo electrónico un comunicado de prensa a una agencia de noticias, en el que declaraba falsamente que Emulex Corporation iba a publicar una gran pérdida trimestral y que el director general iba a renunciar de inmediato. En pocas horas, las acciones de la compañía cayeron un 60%, lo cual provocó que los inversionistas perdieran más de $2 mil millones. El autor de esta noticia ganó un cuarto de millón de dólares al vender sus acciones poco antes de enviar el anuncio. Si bien este evento no tuvo que ver con que alguien irrumpiera en un sitio web, está claro que colocar un anuncio de tal magnitud en una página de inicio de cualquier corporación importante podría tener un efecto similar.

Podríamos (por desgracia) seguir contando muchos casos como éstos, pero es tiempo de que examinemos algunos de los aspectos técnicos relacionados con la seguridad en la web. Para mayor información sobre los problemas de seguridad de todo tipo, consulte a Anderson (2008a), Stuttard y Pinto (2007), y Schneier (2004). Si busca en Internet también encontrará una gran cantidad de casos específicos.

8.9.2 Asignación segura de nombres

Empecemos con algo muy básico: Alice desea visitar el sitio web de Bob. Ella teclea el URL de él en su navegador y segundos más tarde, aparece una página web. Pero, ¿es la de Bob? Tal vez. Trudy podría estar haciendo sus viejos trucos nuevamente. Por ejemplo, podría estar interceptando y examinando todos los paquetes salientes de Alice. Si captura una solicitud *GET* de HTTP dirigida al sitio web de Bob, podría ir ella misma a dicho sitio para obtener la página, modificarla como lo desea y regresar la página falsa a Alice, quien no se daría cuenta de ello. Peor aún, Trudy podría recortar los precios de la tienda electrónica de Bob para hacer que parezcan muy atractivos, con lo que engañaría a Alice para que enviara su número de tarjeta de crédito a “Bob” para comprar algo de mercancía.

Una desventaja de este clásico ataque de intermediario es que Trudy tiene que estar en una posición para interceptar el tráfico saliente de Alice y falsificar su tráfico entrante. En la práctica, tiene que intervenir la línea telefónica de Alice o la de Bob, puesto que intervenir la red troncal de fibra es mucho más difícil. Aunque intervenir la línea es posible, también significa mucho trabajo, y si bien Trudy es inteligente, también es floja. Además, hay formas más fáciles de engañar a Alice.

Falsificación de DNS

Una forma sería que Trudy violara el sistema DNS, o tal vez sólo el caché DNS del ISP de Alice, y reemplazara la dirección IP de Bob (por decir, 36.1.2.3) con su propia dirección IP (por decir, 42.9.9.9). Esto nos lleva al siguiente ataque. En la figura 8-46(a) se ilustra la forma en que se supone debe trabajar.

Aquí, Alice (1) pide al DNS la dirección IP de Bob, (2) la obtiene, (3) le pide a él su página de inicio y (4) también la obtiene. Después de que Trudy ha modificado el registro DNS de Bob para que contenga su propia dirección IP en lugar de la de Bob, obtenemos la situación de la figura 8-46(b). Aquí, cuando Alice busca la dirección IP de Bob, obtiene la de Trudy, por lo que todo su tráfico dirigido a Bob va a parar a las manos de Trudy. Ahora, Trudy puede iniciar un ataque de intermediario sin tener que intervenir ninguna línea telefónica. En cambio, tiene que irrumpir en un servidor DNS y cambiar un registro, lo cual es mucho más fácil.

¿Cómo podría Trudy engañar al DNS? Resulta ser muy sencillo. En esencia, Trudy puede engañar al servidor DNS en el ISP de Alice para que envíe una consulta con el fin de encontrar la dirección de Bob. Por desgracia, puesto que el DNS utiliza UDP, el servidor DNS no tiene una verdadera forma de verificar quién proporcionó la respuesta. Trudy puede explotar esta propiedad, para lo cual falsifica la respuesta esperada e inyecta una dirección IP falsa en el caché del servidor DNS. Para simplificar, supondremos que el ISP de Alice no tiene en un principio una entrada para el sitio web de Bob, *bob.com*. Si la tiene, Trudy puede esperar hasta que expire y probar otra vez (o utilizar otros trucos).

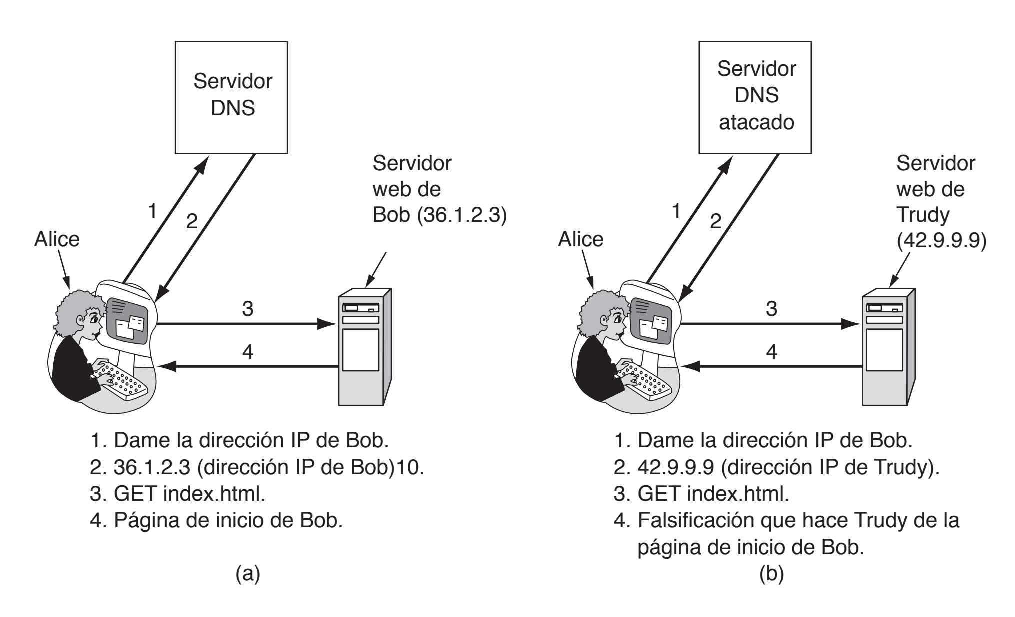


Figura 8-46. (a) Situación normal. (b) Un ataque basado en irrumpir en un servidor DNS para modificar el registro de Bob.

Para iniciar el ataque, Trudy envía una solicitud de búsqueda al ISP de Alice y le pide la dirección IP de *bob.com*. Puesto que no hay entrada para este nombre DNS, el servidor de caché consulta con el servidor de nivel superior el dominio *com* para obtener uno. Sin embargo, Trudy le gana al servidor *com* y regresa una respuesta falsa en la que dice: "*bob.com* es 42.9.9.9", y esa dirección IP es la de ella. Si la respuesta falsa de Trudy llega primero al ISP de Alice, se almacenará en caché y la respuesta real se rechazará como respuesta no solicitada de una consulta que ya no está pendiente. A la acción de engañar a un servidor DNS para que instale una dirección IP falsa se le conoce como **falsificación de DNS**. Una caché que contiene una dirección IP intencionalmente falsa como ésta se conoce como **caché envenenada**.

En realidad, las cosas no son tan sencillas. En primer lugar, el ISP de Alice verifica si la respuesta lleva la dirección de origen IP correcta del servidor de nivel superior. Pero como Trudy puede colocar lo que

quiera en ese campo IP, puede vencer esa prueba con facilidad ya que las direcciones IP de los servidores de nivel superior tienen que ser públicas.

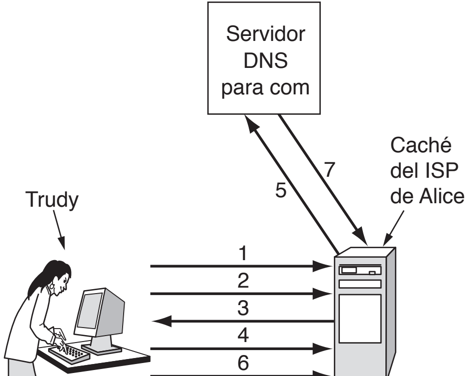
En segundo lugar, para permitir que los servidores DNS digan cuál respuesta corresponde a cuál solicitud, todas las solicitudes llevan un número de secuencia. Para falsificar el ISP de Alice, Trudy tiene que conocer su número de secuencia actual. La forma más fácil para que Trudy lo conozca es que ella misma registre un dominio, digamos, *trudy-la-intrusa.com*. Supongamos que su dirección IP también es 42.9.9.9. Trudy también crea un servidor DNS para su nuevo dominio, *dns.trudy-la-intrusa.com*. Este servidor DNS también utiliza la dirección IP 42.9.9.9 de Trudy, puesto que ésta sólo tiene una computadora. Ahora Trudy tiene que informarle al ISP de Alice sobre su servidor DNS. Esto es fácil. Todo lo que tiene que hacer es pedir *loquesea.trudy-la-intrusa.com* al ISP de Alice, lo que causará que dicho ISP pregunte al servidor *com* de nivel superior para averiguar quién sirve el nuevo dominio de Trudy.

Una vez que el nombre *dns.trudy-la-intrusa.com* está seguro en la caché del ISP de Alice, puede empezar el verdadero ataque. Ahora Trudy solicita *www.trudy-la-intrusa.com* al ISP de Alice. Como es natural, el ISP envía al servidor DNS de Trudy una consulta en la que le pide ese nombre. Dicha consulta lleva el número de secuencia que Trudy está buscando. Rápidamente, Trudy pide al ISP de Alice que busque a Bob. Trudy responde de inmediato su propia pregunta y envía al ISP una respuesta falsificada, supuestamente proveniente del servidor *com* de nivel superior, que dice: “*bob.com* es 42.9.9.9”. Esta respuesta falsificada lleva un número de secuencia que es un número mayor que el que acaba de recibir. Ya que está allí, puede enviar una segunda falsificación con un número de secuencia aumentado en dos, y tal vez una docena más con números de secuencia cada vez mayores. Uno de ellos deberá coincidir. El resto simplemente se elimina. Cuando llega la respuesta falsificada de Alice, se almacena en caché; cuando más tarde llega la respuesta real, se rechaza puesto que no hay ninguna consulta pendiente.

Ahora, cuando Alice busca a *bob.com*, se le indica que utilice 42.9.9.9, la dirección de Trudy. Ésta ha montado un ataque de intermediario exitoso desde la comodidad de su hogar. En la figura 8-47 se ilustran los diversos pasos para este ataque. Para frustrar este ataque específico, hay que tener varios servidores DNS que usen números ID aleatorios en sus consultas en vez de sólo contar, pero parece que cada vez que se tapa un orificio, aparece otro. En específico, los números ID son sólo de 16 bits, por lo que es fácil recorrerlos todos si se utiliza una computadora.

DNS seguro

El verdadero problema es que el DNS se diseñó en la época en la que Internet era una herramienta de investigación para unos cuantos cientos de universidades y ni Alice, ni Bob, y mucho menos Trudy, fueron



1. Busca foobar.trudy-la-intrusa.com (para obligarlo a entrar al caché del ISP).
2. Busca www.trudy-la-intrusa.com (para obtener el siguiente número de secuencia del ISP).
3. Solicita www.trudy-la-intrusa.com (transporta el siguiente número de secuencia del ISP, n).
4. Con la rapidez de un conejo, busca bob.com (para obligar a que el ISP consulte al servidor com en el paso 5).
5. Consulta legítima para bob.com con sec = n+1.
6. Respuesta falsificada de Trudy: Bob es 42.9.9.9, sec = n+1.
7. Respuesta real (rechazada, demasiado tarde).

Figura 8-47. La forma en que Trudy falsifica el ISP de Alice.

invitados a la fiesta. En ese entonces la seguridad no era un problema; hacer que Internet funcionara era el problema. Con los años, el entorno ha cambiado en forma radical, por lo que en 1994 la IETF estableció un grupo funcional para hacer que DNS fuera básicamente seguro. Este proyecto (que continúa) se conoce como **DNSsec** (**Seguridad DNS**, del inglés *DNS security*); su primer resultado se presentó en el RFC 2535. Por desgracia, DNSsec aún no se ha implementado por completo, por lo que hay muchos servidores DNS que aún son vulnerables a los ataques de falsificación.

En concepto, DNSsec es muy simple. Se basa en la criptografía de clave pública. Cada zona DNS (en el sentido de la figura 7-5) tiene un par de claves pública/privada. Toda la información enviada por un servidor DNS se firma con la clave privada de la zona originaria, por lo que el receptor puede verificar su autenticidad.

DNSsec ofrece tres servicios fundamentales:

1. Prueba del origen de los datos.
2. Distribución de la clave pública.
3. Autentificación de transacciones y solicitudes.

El servicio principal es el que se lista primero, el cual verifica que los datos que se regresan hayan sido aprobados por el propietario de la zona. El segundo es útil para almacenar y recuperar claves públicas en forma segura. El tercero es necesario para protegerse contra ataques de repetición y falsificación. Observe que la confidencialidad no se ofrece como servicio, puesto que toda la información en el DNS se considera pública. Como se espera que la planificación en etapas del DNSsec se lleve algunos años, es esencial la capacidad que tienen los servidores conscientes de la seguridad para interactuar con los servidores que no están conscientes de ella, lo cual implica que no se puede cambiar el protocolo. Veamos ahora algunos de los detalles.

Los registros DNS están agrupados en conjuntos llamados **RRSets** (**Conjuntos de Registros de Recursos**, del inglés *Resource Record Sets*), en los que todos los registros que tienen el mismo nombre, clase y tipo se agrupan en un conjunto. Por ejemplo, un RRSet puede contener varios registros *A* si un nombre DNS se resuelve en una dirección IP primaria y una dirección IP secundaria. Los RRSets se extienden con varios nuevos tipos de registros (que veremos a continuación). A cada RRSet se le aplica un hash de manera criptográfica (por ejemplo, usando SHA-1). La clave privada de la zona firma el hash (por ejemplo, mediante RSA). La unidad de transmisión a los clientes es el RRSet firmado. Al momento de recibir un RRSet firmado, el cliente puede verificar si fue firmado o no por la clave privada de la zona originaria. Si la firma concuerda, se aceptan los datos. Como cada RRSet contiene su propia firma, los RRSets se pueden almacenar en caché en cualquier lugar, incluso en servidores no confiables, sin poner en peligro la seguridad.

DNSsec introduce varios tipos nuevos de registros. El primero de éstos es el registro *KEY*. Este registro mantiene la clave pública de una zona, usuario, host u otro personaje principal, el algoritmo criptográfico utilizado para firmar, el protocolo utilizado para la transmisión y unos cuantos bits más. La clave pública se almacena tal como está. Los certificados X.509 no se utilizan debido a su volumen. El campo de algoritmo contiene un 1 para las firmas MD5/RSA (la opción preferida) y otros valores para otras combinaciones. El campo de protocolo puede indicar el uso de IPsec y otros protocolos de seguridad, si los hay.

SIG es el segundo nuevo tipo de registro. Contiene el hash firmado de acuerdo con el algoritmo especificado en el registro *KEY*. La firma se aplica a todos los registros del RRSet, incluyendo cualquiera de los registros *KEY* presentes, pero excluyéndose a sí mismo. También contiene los tiempos en los que la firma comienza su periodo de validación y cuando ésta expira, así como el nombre de quien firma y algunos elementos más.

El diseño de DNSsec tiene la característica de que es posible mantener sin conexión la clave privada de una zona. Una o dos veces al día, el contenido de una base de datos de una zona se puede transportar

en forma manual (por ejemplo, en CD-ROM) a una máquina desconectada en la que se localiza una clave privada. Todos los RRSets se pueden firmar ahí y los registros *SIG* que se producen de esa manera se pueden transportar de vuelta al servidor primario de la zona en CD-ROM. Así, es posible almacenar la clave privada en un CD-ROM encerrado en una caja fuerte, excepto cuando se inserta en la máquina desconectada para firmar los nuevos RRSets del día. Al terminar el proceso de firma, todas las copias de la clave se eliminan de la memoria y tanto el disco como el CD-ROM se regresan a la caja fuerte. Este procedimiento reduce la seguridad electrónica a seguridad física, algo con lo que las personas saben lidiar.

Este método de firmar antes los RRSets aumenta de manera considerable la velocidad del proceso de responder a las consultas, puesto que no es necesario realizar criptografía sobre la marcha. La desventaja es que se necesita una gran cantidad de espacio en disco para almacenar todas las claves y firmas en las bases de datos del DNS. Algunos registros incrementarán hasta 10 veces su tamaño debido a la firma.

Cuando un proceso cliente recibe un RRSet firmado, éste debe aplicar la clave pública de la zona originaria para desencriptar el hash, calcular el hash mismo y comparar los dos valores. Si concuerdan, los datos se consideran válidos. Sin embargo, este procedimiento trae consigo la pregunta de cómo obtiene el cliente la clave pública de la zona. Una forma es adquirirla de un servidor de confianza, mediante el uso de una conexión segura (por ejemplo, mediante IPsec).

Sin embargo, en la práctica se espera que los clientes se configuren previamente con las claves públicas de todos los dominios de nivel superior. Si Alice ahora desea visitar el sitio web de Bob, puede pedir al DNS el RRSet de *bob.com*, el cual contendrá su dirección IP y un registro *KEY* que contiene la clave pública de Bob. Este RRSet será firmado por el dominio *com* de nivel superior, por lo que Alice puede verificar fácilmente su validez. Un ejemplo de lo que podría contener este RRSet se muestra en la figura 8-48.

Nombre del dominio	Tiempo de vida	Clase	Tipo	Valor
bob.com.	86400	IN	A	36.1.2.3
bob.com.	86400	IN	KEY	3682793A7B73F731029CE2737D…
bob.com.	86400	IN	SIG	86947503A8B848F5272E53930C…

Figura 8-48. Ejemplo de un RRSet para *bob.com*. El registro *KEY* es la clave pública de Bob. El registro *SIG* es el hash firmado del servidor *com* de nivel superior de los registros *A* y *KEY* para verificar su autenticidad.

Una vez que Alice tiene una copia verificada de la clave pública de Bob, puede pedir al servidor DNS (operado por Bob) la dirección IP de *www.bob.com*. La clave privada de Bob firmará este RRSet, con el fin de que Alice pueda verificar la firma del RRSet que Bob regresa. Si de alguna forma Trudy logra inyectar un RRSet falso en cualquiera de las cachés, Alice puede detectar fácilmente su falta de autenticidad porque el registro *SIG* que contenga será incorrecto.

Sin embargo, DNSsec también proporciona un mecanismo criptográfico para enlazar una respuesta a una consulta específica, con el fin de evitar el tipo de falsificación utilizado por Trudy en la figura 8-47. Esta medida (opcional) de antifalsificación agrega a la respuesta un hash del mensaje de consulta firmado con la clave privada del receptor. Puesto que Trudy no conoce la clave privada del servidor *com* de nivel superior, no puede falsificar una respuesta a una consulta que el ISP de Alice haya enviado a dicho servidor. Ciertamente Trudy puede conseguir que su respuesta regrese primero, pero será rechazada debido a la firma inválida sobre la consulta a la que se le aplicó el hash.

DNSsec también soporta unos cuantos tipos de registros más. Por ejemplo, el registro *CERT* se puede utilizar para almacenar certificados (por ejemplo, X.509). Se proporcionó este registro porque algunas personas

desean convertir el DNS en una PKI. Aún no se sabe si esto realmente ocurrirá o no. Detendremos nuestro análisis de DNSsec aquí. Para mayores detalles, por favor consulte el RFC 2535.

8.9.3 SSL: la capa de sockets seguros

La asignación segura de nombres es un buen inicio, pero hay mucho más sobre la seguridad en la web. El siguiente paso son las conexiones seguras. A continuación veremos cómo puede lograrse este tipo de conexiones. Nada que involucre a la seguridad es simple, y esto no es la excepción.

Cuando la web irrumpió a la vista pública, en un principio sólo se utilizaba para distribuir páginas estáticas. Sin embargo, pronto algunas compañías tuvieron la idea de utilizarla para transacciones financieras, como para comprar mercancía con tarjeta de crédito, operaciones bancarias en línea y comercio electrónico de acciones. Estas aplicaciones crearon una demanda de conexiones seguras. En 1995, Netscape Communications Corp., que entonces dominaba el mercado de los navegadores, respondió a esta demanda mediante la introducción de un paquete de seguridad llamado **SSL** (**Capa de Sockets Seguros**, del inglés *Secure Sockets Layer*). En la actualidad, este software y su protocolo se utilizan mucho, por ejemplo, en Firefox, Safari e Internet Explorer, por lo que vale la pena examinarlo con cierto detalle.

SSL construye una conexión segura entre dos sockets, incluyendo:

1. Negociación de parámetros entre el cliente y el servidor.
2. Autentificación del servidor por el cliente.
3. Comunicación secreta.
4. Protección de la integridad de los datos.

Ya hemos visto antes estos elementos, por lo que no hay necesidad de explicarlos de nuevo.

En la figura 8-49 se ilustra la posición de SSL en la pila de protocolos usuales. En efecto, es una nueva capa interpuesta entre la capa de aplicación y la capa de transporte, que acepta solicitudes del navegador y las envía a TCP para transmitirlas al servidor. Una vez que se ha establecido la conexión segura, el trabajo principal de SSL es manejar la compresión y encriptación. Cuando se utiliza HTTP sobre SSL, se conoce como **HTTPS** (**HTTP Seguro**, del inglés *HTTP Secure*), aunque sea el protocolo HTTP estándar. Algunas veces está disponible en un nuevo puerto (443) en lugar del puerto 80. Además, SSL no está restringido a utilizarse sólo con navegadores web, pero ésa es su aplicación más común. También puede proporcionar autentificación mutua.

Aplicación (HTTP)
Seguridad (SSL)
Transporte (TCP)
Red (IP)
Enlace de datos (PPP)
Física (módem, ADSL, TV por cable)

Figura 8-49. Capas (y protocolos) para un usuario doméstico que navega con SSL.

El protocolo SSL ha pasado por varias versiones. A continuación sólo examinaremos la versión 3, que es la versión más utilizada en la actualidad. SSL soporta una variedad de opciones diferentes. Entre dichas opciones se incluyen la presencia o ausencia de compresión, los algoritmos criptográficos a utilizar y

algunos asuntos relacionados con las restricciones de exportación sobre la criptografía. La última se destina principalmente para asegurarse de que se utilice criptografía seria sólo cuando ambos lados de la conexión estén en Estados Unidos. En otros casos, las claves se limitan a 40 bits, lo que los criptógrafos consideran como una broma. El gobierno de Estados Unidos obligó a Netscape a incluir esta restricción para obtener una licencia de exportación.

SSL está formado de dos subprotocolos, uno para establecer una conexión segura y otro para utilizarla. Para empezar, vamos a ver cómo se establecen las conexiones seguras. En la figura 8-50 se muestra el subprotocolo de establecimiento de conexión. Comienza con el mensaje 1, cuando Alice envía una solicitud a Bob para establecer una conexión. La solicitud especifica la versión SSL que tiene Alice y sus preferencias con respecto a los algoritmos criptográficos y de compresión. También contiene un nonce R_A, para utilizarlo más tarde.

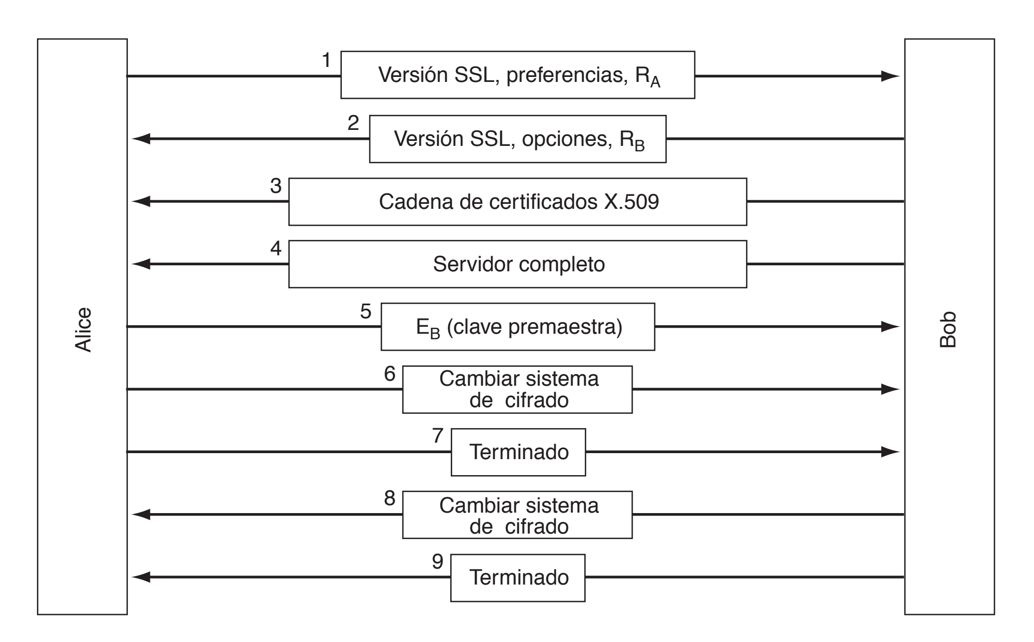


Figura 8-50. Una versión simplificada del subprotocolo de establecimiento de conexión SSL.

Ahora es el turno de Bob. En el mensaje 2, Bob selecciona uno de los diversos algoritmos que Alice puede soportar y envía su propio nonce, R_B. Después, en el mensaje 3 envía un certificado que contiene su clave pública. Si este certificado no está firmado por alguna autoridad bien conocida, también envía una cadena de certificados que pueden seguirse hasta encontrar uno. Todos los navegadores, incluyendo el de Alice, vienen precargados con cerca de 100 claves públicas, por lo que si Bob puede establecer una cadena anclada a una de ellas, Alice podrá verificar la clave pública de Bob. En este punto Bob podría enviar algunos mensajes más (por ejemplo, podría solicitar el certificado de la clave pública de Alice). Cuando Bob termina, envía el mensaje 4 para indicar a Alice que es su turno.

Para responder, Alice selecciona una clave premaestra aleatoria de 384 bits y la envía a Bob encriptada con la clave pública de él (mensaje 5). La clave de sesión real utilizada para encriptar datos se deriva de la clave premaestra combinada con los dos nonces de una forma compleja. Después de recibir el mensaje 5, tanto Alice como Bob pueden calcular la clave de sesión. Por esta razón, Alice indica a Bob que cambie al nuevo sistema de cifrado (mensaje 6) y también que ha terminado con el subprotocolo de establecimiento (mensaje 7). Después Bob confirma que ha recibido la indicación (mensajes 8 y 9).

Sin embargo, aunque Alice sabe quién es Bob, éste no sabe quién es Alice (a menos que ella tenga una clave pública y el correspondiente certificado para ella, una situación poco probable para un individuo). Por lo tanto, el primer mensaje de Bob puede ser una solicitud para que Alice inicie una sesión utilizando un nombre de inicio de sesión y una contraseña previamente establecidos. Sin embargo, el protocolo de inicio de sesión está fuera del alcance de SSL. Una vez iniciada la sesión por cualquier medio, puede comenzar el transporte de los datos.

Como se mencionó antes, SSL soporta múltiples algoritmos criptográficos. El más robusto utiliza triple DES con tres claves separadas para encriptación y SHA-1 para la integridad de los mensajes. Esta combinación es relativamente lenta, por lo que se utiliza principalmente para operaciones bancarias y otras aplicaciones en las que se requiere la seguridad de mayor nivel. Para las aplicaciones comunes de comercio electrónico, se utiliza RC4 con una clave de 128 bits para encriptación y MD5 para la autentificación de mensajes. RC4 toma la clave de 128 bits como semilla y la expande a un número mucho más grande para uso interno. Después utiliza este número interno para generar un flujo de claves. A éste se le aplica un OR exclusivo con el texto plano para proporcionar un sistema de cifrado clásico de flujo, como vimos en la figura 8-14. Las versiones de exportación también utilizan RC4 con claves de 128 bits, 88 de los cuales se hacen públicos para que el sistema de cifrado sea fácil de quebrantar.

Para un transporte real se utiliza un segundo subprotocolo, como se muestra en la figura 8-51. Los mensajes que provengan del navegador primero se dividen en unidades de hasta 16 KB. Si se activa la compresión de datos, cada unidad se comprime por separado. Después de eso, se obtiene una clave secreta a partir de los dos nonces y la clave premaestra se concatena con el texto comprimido; después, al resultado se le aplica un hash con el algoritmo de hash acordado (por lo general MD5). Este hash se adjunta a cada fragmento como el MAC. Después, el fragmento comprimido y el MAC se encriptan con el algoritmo de encriptación simétrico acordado (por lo general, se le aplica un OR exclusivo con el flujo de claves RC4). Por último, se adjunta un encabezado de fragmento y el fragmento se transmite a través de la conexión TCP.

Sin embargo, es necesaria una advertencia. Puesto que se ha mostrado que el RC4 tiene claves débiles que se pueden criptoanalizar con facilidad, la seguridad de SSL mediante RC4 no es muy confiable (Fluhrer y colaboradores, 2001). Los navegadores que permiten al usuario elegir el conjunto de sistema de cifrado se deben configurar para utilizar todo el tiempo triple DES con claves de 168 bits y SHA-1, aun

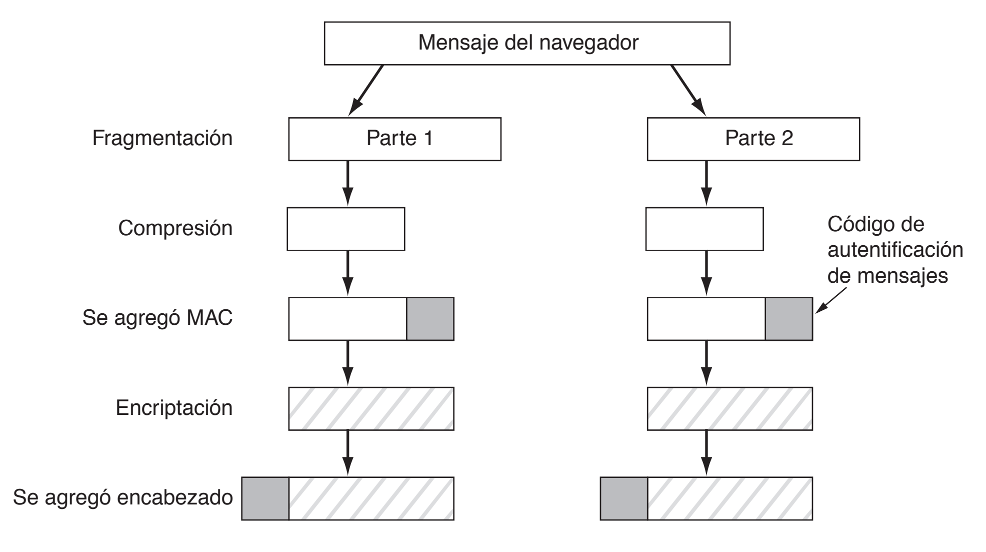


Figura 8-51. Transmisión de datos mediante SSL.

cuando esta combinación sea más lenta que RC4 y MD5. O mejor aún, los usuarios deberían actualizar a navegadores que soporten el sucesor de SSL, que veremos en breve.

Otro problema con SSL es que tal vez los personajes principales no tengan certificados, e incluso si los tienen, no siempre verifican que coincidan las claves que se utilizan.

En 1996, Netscape Communications Corp. entregó el SSL a la IETF para su estandarización. El resultado fue **TLS** (**Seguridad de la Capa de Transporte**, del inglés *Transport Layer Security*); se describe en el RFC 5246.

TLS se basa en la versión 3 de SSL. Los cambios en SSL fueron relativamente pequeños, pero son suficientes como para que SSL versión 3 y TLS no puedan interoperar. Por ejemplo, la forma en que se obtiene una clave de sesión a partir de la clave premaestra y los nonces se cambió para hacer que la clave fuera más fuerte (es decir, más difícil de criptoanalizar). Debido a esta incompatibilidad, la mayoría de los navegadores implementan ambos protocolos, en donde TLS recurre a SSL durante la negociación si es necesario. A esto se le conoce como SSL/TLS. La primera implementación de TLS apareció en 1999, y la versión 1.2 se definió en agosto de 2008. Incluye soporte para suites de cifrado más robustas (en especial AES). SSL ha permanecido sólido en el mercado, aunque es probable que TLS lo reemplace en forma gradual.

8.9.4 Seguridad de código móvil

La asignación de nombres y las conexiones son dos áreas de preocupación que se relacionan con la seguridad en web. Pero hay más. En los primeros días, cuando las páginas web eran simplemente archivos HTML estáticos, no contenían código ejecutable. Ahora es común que contengan pequeños programas, incluyendo applets de Java, controles ActiveX y secuencias de JavaScript. Es obvio que descargar y ejecutar este tipo de **código móvil** es un riesgo de seguridad masivo, por lo que se han diseñado varios métodos para minimizarlo. A continuación daremos un vistazo rápido a varios de los problemas surgidos debido al código móvil y algunos métodos para lidiar con ellos.

Seguridad de los applets de Java

Los applets de Java son pequeños programas de Java compilados a un lenguaje de máquina orientado a pilas, conocido como **JVM** (**Máquina Virtual de Java**, del inglés *Java Virtual Machine*). Se pueden colocar en una página web para descargarlos junto con dicha página. Una vez que se carga la página, las applets se insertan en un intérprete JVM dentro del navegador, como se ilustra en la figura 8-52.

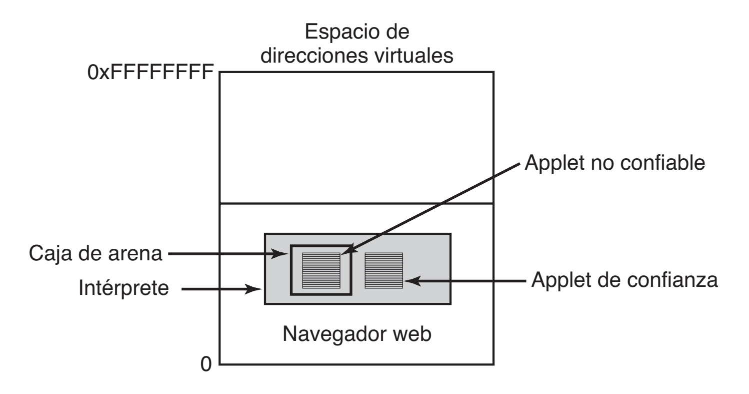


Figura 8-52. Las applets se pueden interpretar mediante un navegador web.

La ventaja de ejecutar código interpretado en lugar de compilado es que el intérprete puede examinar cada instrucción antes de ejecutarla. Esto da al intérprete la oportunidad de verificar si la dirección de la instrucción es válida. Además, las llamadas de sistema también se atrapan e interpretan. La forma en que se manejan estas llamadas depende de la política de seguridad. Por ejemplo, si una applet es de confianza (por decir, que provenga del disco local), sus llamadas de sistema podrían llevarse a cabo sin cuestionamiento. No obstante, si un applet no es confiable (por ejemplo, proviene de Internet), se podría encapsular en lo que se conoce como una **caja de arena** (*sandbox*) para restringir su comportamiento y atrapar sus intentos por utilizar los recursos del sistema.

Cuando una applet intenta utilizar un recurso del sistema, su llamada se pasa al monitor de seguridad para que la apruebe. El monitor examina la llamada a la luz de la política de seguridad local y después toma una decisión para permitirla o rechazarla. De esta manera, es posible proporcionar acceso a los applets a algunos recursos, pero no a todos. Por desgracia, la realidad es que el modelo de seguridad no funciona del todo bien y surgen errores con frecuencia.

ActiveX

Los controles ActiveX son programas binarios x86 que se pueden incrustar en páginas web. Al encontrarnos uno de ellos, se realiza una verificación para ver si se debe ejecutar y, si pasa la prueba, se ejecuta. No se interpreta ni se pone en una caja de arena de ninguna manera, por lo que tiene tanto poder como cualquier otro programa de usuario, y puede llegar a hacer mucho daño. Por lo tanto, toda la seguridad recae en la decisión de si se ejecuta o no el control ActiveX. En retrospectiva, toda la idea es un gigantesco hoyo de seguridad.

El método que Microsoft eligió para tomar esta decisión se basa en la idea de la **firma de código**. Cada control ActiveX está acompañado por una firma digital —un hash del código que está firmado por su creador mediante el uso de la criptografía de clave pública—. Cuando aparece un control ActiveX, el navegador primero verifica la firma para asegurarse de que no se haya alterado en el camino. Si la firma es correcta, el navegador verifica a continuación sus tablas internas para ver si el creador del programa es confiable o si hay una cadena de confianza hacia un creador confiable. Si el creador es confiable, el programa se ejecuta; de lo contrario, no se ejecuta. El sistema de Microsoft para verificar controles ActiveX se conoce como **Authenticode**.

Es útil comparar los métodos de Java y ActiveX. Con el método de Java, no se realiza ningún intento por determinar quién escribió el applet. En cambio, un intérprete en tiempo de ejecución se asegura de que la applet no haga lo que el dueño de la máquina haya prohibido. En contraste, con la firma de código no hay intento por supervisar el comportamiento en tiempo de ejecución del código móvil. Si proviene de un origen confiable y no se ha modificado en el trayecto, simplemente se ejecuta. No se realiza ningún intento por ver si el código es malicioso o no. Si el programador original *destinó* el código para dar formato al disco duro y después borra la flash ROM con el fin de que la computadora nunca se vuelva a encender, y si el programador ha sido certificado como confiable, el código se ejecutará y destruirá la computadora (a menos que se hayan desactivado los controles ActiveX en el navegador).

Muchas personas creen que no es conveniente confiar en software de una compañía desconocida. Para demostrar el problema, un programador de Seattle formó una compañía de software y la certificó como confiable, lo cual es fácil de hacer. Después escribió un control ActiveX que apagaba de manera instantánea la máquina y lo distribuyó para red. Apagó muchas máquinas, pero podían encenderse otra vez, por lo que no causó ningún daño. Simplemente trataba de exponer el problema al mundo. La respuesta oficial fue revocar el certificado a este control ActiveX específico, lo cual acabó con un corto episodio de profunda vergüenza, pero el problema subyacente aún está allí para que un programador malicioso se aproveche de él (Garfinkel con Spafford, 2002). Puesto que no hay forma de vigilar las miles de compañías de software que podrían escribir código móvil, la técnica de firma de código es un desastre al acecho.

JavaScript

JavaScript no tiene ningún modelo de seguridad formal, pero tiene un largo historial de implementaciones defectuosas. Cada fabricante maneja la seguridad de diferente forma. Por ejemplo, la versión 2 de Netscape Navigator utilizaba algo semejante al modelo de Java, pero en la versión 4 se abandonó por un modelo de firma de código.

El problema fundamental es que permitir la ejecución de código extraño en su máquina es abrir la puerta a los problemas. Desde el punto de vista de la seguridad, es como invitar a un ladrón a su casa y después tratar de vigilarlo con cuidado de manera que no pueda ir de la cocina a la sala. Si ocurre algo inesperado y usted está distraído por un momento, pueden suceder cosas malas. El suspenso aquí es que el código móvil permite gráficos vistosos e interacción rápida, y muchos diseñadores de sitios web piensan que esto es más importante que la seguridad, en especial cuando es la máquina de alguien más la que está en riesgo.

Extensiones del navegador

Además de extender las páginas web con código, también hay auge en el mercado por las **extensiones** y **complementos** (*add-ons* o *plug-ins*) **del navegador**. Éstos son programas de computadora que extienden la funcionalidad de los navegadores web. A menudo, los *plug-ins* proveen la capacidad de interpretar o visualizar cierto tipo de contenido, como los archivos PDF o las animaciones en Flash. Las extensiones y los *add-ons* proveen nuevas características para el navegador, como una mejor administración de las contraseñas, o formas de interactuar con las páginas al marcarlas o activar la compra rápida para artículos relacionados, por ejemplo.

La instalación de una extensión, *add-on* o *plug-in* es tan simple como toparnos con algo que nos gusta al navegar y seguir el vínculo para instalar el programa. Esta acción provocará la descarga de código a través de Internet, el cual se instalará en el navegador. Todos estos programas se escriben en marcos de trabajo que difieren según el navegador en el que se vayan a aplicar. Sin embargo, como una primera aproximación, ellos se vuelven parte de la base de cómputo de confianza del navegador. Es decir, si el código que se instala tiene errores, todo el navegador puede verse comprometido.

También hay otros dos modos obvios de falla. El primero es que el programa se puede comportar en forma maliciosa; por ejemplo, al recopilar información personal y enviarla a un servidor remoto. En lo que al navegador respecta, el usuario instaló la extensión precisamente para este fin. El segundo problema es que los *plug-ins* otorgan al navegador la habilidad de interpretar nuevos tipos de contenido. A menudo este contenido es todo un lenguaje de programación completo. PDF y Flash son buenos ejemplos de esto. Cuando los usuarios ven páginas con contenido PDF y Flash, los *plug-ins* en sus navegadores ejecutan el código PDF y Flash. Más vale que ese código sea seguro; con frecuencia hay vulnerabilidades que se pueden explotar. Por todas estas razones, sólo hay que instalar *add-ons* y *plug-ins* cuando sea necesario, y sólo de proveedores de confianza.

Virus

Los virus son otra forma de código móvil. Sólo que a diferencia de los ejemplos anteriores, los virus no se invitan de ninguna manera. La diferencia entre un virus y el código móvil ordinario es que los virus se reproducen por sí mismos. Cuando llega uno, ya sea por medio de una página web, un archivo adjunto de correo electrónico o por algún otro medio, lo común es que empiecen por infectar los programas ejecutables del disco. Cuando se ejecuta alguno de estos programas, el control se transfiere al virus, que por lo general trata de esparcirse a sí mismo a otras máquinas; por ejemplo, puede enviar por

correo electrónico copias de sí mismo a cualquier persona que se encuentre en la libreta de direcciones de la víctima. Algunos virus infectan el sector de arranque del disco duro, por lo que cuando la máquina se inicia, el virus se ejecuta. Los virus se han convertido en un gran problema en Internet y han causado que se pierdan miles de millones de dólares por los daños. No hay una solución obvia a este problema. Tal vez podría ayudar una nueva generación de sistemas operativos basados en microkernels seguros y la compartimentación limitada de usuarios, procesos y recursos.

8.10 ASPECTOS SOCIALES

Internet y su tecnología de seguridad es un área donde confluyen los aspectos sociales, la política pública y la tecnología, a menudo con grandes consecuencias. A continuación sólo examinaremos con brevedad tres áreas: privacidad, libertad de expresión y derechos de autor. No es necesario decir que sólo podremos arañar la superficie del tema. Para mayor información consulte a Anderson (2008a), Garfinkel con Spafford (2002) y Schneier (2004). En Internet también puede encontrar mucho material. Simplemente introduzca en cualquier motor de búsqueda palabras como "privacidad", "censura" y "derechos de autor". Además, vea el sitio web de este libro para obtener algunos vínculos. Está en *http://www.pearsonhighered.com/tanenbaum*.

8.10.1 Privacidad

¿Tienen las personas derecho a la privacidad? Buena pregunta. La Cuarta Enmienda de la Constitución de Estados Unidos prohíbe que el gobierno busque en las casas, documentos y bienes de las personas sin una razón válida y restringe las circunstancias en las que se deben emitir las órdenes de cateo. Por lo tanto, la privacidad ha estado en la agenda pública durante casi 200 años, por lo menos en Estados Unidos.

En la década pasada cambió la facilidad con la que el gobierno podía espiar a sus ciudadanos y la facilidad con la que éstos evitaban tal espionaje. En el siglo XVIII, para que el gobierno pudiera buscar en los documentos de los ciudadanos, tenía que enviar un policía a caballo a la granja de dicho ciudadano para que le exigiera ver ciertos documentos. Era un procedimiento engorroso. Hoy en día, las compañías telefónicas y los proveedores de Internet proporcionan con facilidad intervenciones telefónicas cuando se les presenta una orden de cateo. Esto facilita la vida del policía y ya no hay peligro de que se caiga del caballo.

La criptografía cambia todo esto. Cualquiera que se tome la molestia de bajar e instalar PGP y que utilice una clave bien custodiada de categoría extraterrestre puede estar seguro de que nadie en el universo conocido podrá leer su correo electrónico, haya o no orden de cateo. Los gobiernos entienden bien esto y no les gusta. La privacidad real para ellos significa que les será mucho más difícil espiar a los criminales de todo tipo, y todavía les será más difícil espiar a los reporteros y oponentes políticos. En consecuencia, algunos gobiernos restringen o prohíben el uso o exportación de la criptografía. Por ejemplo, en Francia, antes de 1999, la criptografía estaba prohibida a menos que se le proporcionaran las claves al gobierno.

Esto no sólo sucedía en Francia. En abril de 1993, el gobierno de Estados Unidos anunció su intención de hacer que un criptoprocesador de hardware, el **chip clipper**, fuera el estándar en toda la comunicación en red. Se decía que esto garantizaría la privacidad de los ciudadanos. También se mencionó que el procesador proporcionaría al gobierno la capacidad de desencriptar todo el tráfico a través de un esquema llamado **depósito de claves** (*key escrow*), el cual permitía que el gobierno accediera a todas las claves. Sin embargo, el gobierno prometió que sólo espiaría cuando tuviera una orden de cateo válida.

No es necesario decir que se generó una gran controversia por esta situación, en la que los activistas de la privacidad condenaban todo el plan y los oficiales de policía la aclamaban. Con el tiempo, el gobierno se retractó y abandonó la idea.

En el sitio web de la fundación Electronic Frontier Foundation, *www.eff.org*, está disponible una gran cantidad de información acerca de la privacidad electrónica.

Retransmisores de correo anónimos

PGP, SSL y otras tecnologías hacen posible que dos partes establezcan una comunicación segura y autentificada, libre de vigilancia e interferencia de terceros. Sin embargo, algunas veces la privacidad se aplica mejor cuando no hay autentificación; es decir, al hacer que la comunicación sea anónima. El anonimato podría ser conveniente para los mensajes punto a punto, grupos de noticias o ambos.

Veamos algunos ejemplos. En primer lugar, los disidentes políticos que viven bajo regímenes autoritarios con frecuencia desean comunicarse de manera anónima para evitar ser encarcelados o asesinados. En segundo lugar, por lo general las acciones ilegales en muchas organizaciones gubernamentales, educativas y corporativas, entre otras, son denunciadas por personas que con frecuencia prefieren permanecer en el anonimato para evitar represalias. En tercer lugar, las personas con creencias religiosas, políticas y sociales impopulares podrían querer comunicarse entre ellas a través del correo electrónico o de grupos de noticias sin exponerse a sí mismas. En cuarto lugar, las personas podrían desear discutir sobre el alcoholismo, las enfermedades mentales, el acoso sexual, el abuso infantil o ser miembros de una minoría perseguida en un grupo de noticias sin tener que hacerse públicos. Por supuesto que existen muchos otros ejemplos más.

Consideremos un ejemplo específico. En la década de 1990 algunos críticos publicaron sus puntos de vista sobre un grupo religioso no tradicional en un grupo de noticias de USENET a través de un **retransmisor de correo anónimo**. Este servidor permitía que los usuarios crearan seudónimos y le enviaran correo electrónico, y después dicho servidor volvía a enviar o publicar tal correo utilizando el pseudónimo creado, de manera que nadie podía saber de dónde provenían realmente los mensajes. Algunas publicaciones revelaron información que el grupo religioso afirmaba eran secretos comerciales y documentos con derechos de autor. Por lo tanto, dicho grupo fue con las autoridades locales y les dijo que sus secretos comerciales habían sido revelados y que sus derechos de autor habían sido violados, los cuales eran delitos en el lugar donde se encontraba el servidor. En consecuencia, se produjo un juicio y el operador del servidor se vio obligado a entregar la información de correspondencia, la cual reveló las verdaderas identidades de las personas que habían realizado las publicaciones de los mensajes. (Por cierto, ésta no fue la primera vez que un grupo religioso no estaba de acuerdo con que alguien revelara sus secretos comerciales: William Tyndale fue quemado en la hoguera en 1536 por traducir la Biblia al inglés).

Un segmento considerable de la comunidad de Internet se indignó por completo debido a esta brecha de confidencialidad. La conclusión a la que todos llegaron es que no sirve de mucho un retransmisor de correo anónimo que almacena una correspondencia entre las direcciones reales de correo electrónico y los seudónimos (el cual se conoce ahora como retransmisor de correo de tipo 1). Este caso estimuló a varias personas a diseñar retransmisores de correo anónimos que pudieran resistir a los ataques de citaciones legales.

Estos nuevos retransmisores de correo, conocidos comúnmente como **retransmisores de correo cypherpunks**, funcionan como se describe a continuación. El usuario produce un mensaje de correo electrónico, completo con encabezados RFC 822 (excepto *From:*, por supuesto), lo encripta con la clave pública del retransmisor del correo y lo envía a éste. Ahí se eliminan los encabezados externos RFC 822, el contenido se desencripta y el mensaje se retransmite. El retransmisor de correo no tiene cuentas ni mantiene registros, por lo que aunque el servidor se confisque después, no contiene ni una huella de los mensajes que han pasado a través de él.

Muchos usuarios que desean el anonimato encadenan sus solicitudes a través de múltiples retransmisores de correo anónimos, como se muestra en la figura 8-53. Aquí, Alice desea enviar a Bob una tarjeta del Día de San Valentín que sea de verdad anónima, por lo que utiliza tres retransmisores de correo. Redacta el mensaje *M* y coloca en él un encabezado que contiene la dirección de correo electrónico de Bob. Después encripta todo el mensaje con la clave pública del retransmisor de correo 3, E_3 (se indica mediante el sombreado horizontal). Para esto anexa al principio un encabezado con la dirección de correo electrónico en texto plano del retransmisor de correo 3. En la figura, este mensaje es el que se muestra entre los retransmisores de correo 2 y 3.

A continuación Alice encripta este mensaje con la clave pública del retransmisor de correo 2, E_2 (se indica mediante el sombreado vertical) y anexa al principio un encabezado en texto plano que contiene la dirección de correo electrónico del retransmisor de correo 2. Este mensaje se muestra en la figura 8-53, entre los retransmisores de correo 1 y 2. Por último, Alice encripta todo el mensaje con la clave pública del retransmisor de correo 1, E_1, y anexa al inicio un encabezado en texto plano con la dirección de correo electrónico del retransmisor de correo 1. En la figura, este mensaje es el que está a la derecha de Alice y también es el que en realidad transmite.

Cuando el mensaje llega al retransmisor de correo 1, el encabezado exterior se elimina. El cuerpo se desencripta y después se envía al retransmisor de correo 2. En los otros dos retransmisores de correo se realizan pasos similares.

Aunque para cualquiera es muy difícil rastrear el mensaje final de regreso a Alice, muchos retransmisores de correo toman precauciones de seguridad adicionales. Por ejemplo, tal vez mantengan los mensajes durante un tiempo aleatorio, agreguen o eliminen basura al final de un mensaje y reordenen los mensajes, todo esto para dificultar que alguien pueda saber cuál mensaje de un retransmisor de correo corresponde a qué entrada, con el fin de frustrar el análisis de tráfico. Para una descripción de este tipo de retransmisor de correo, consulte a Mazières y Kaashoek (1998).

El anonimato no se limita al correo electrónico. También existen servicios que permiten la navegación anónima en la web mediante el uso de la misma forma de ruta en capas, en la que un nodo sólo conoce al siguiente nodo de la cadena. A este método se le conoce como **enrutamiento cebolla** (*onion routing*), ya que cada nodo tiene que "pelar" otra capa de la cebolla para determinar a dónde debe reenviar el paquete. El usuario configura su navegador para utilizar el servicio generador de anonimato (*anonymizer*) como un proxy. Tor es un ejemplo muy conocido de dicho sistema (Dingledine y colaboradores, 2004). De ahí en adelante, todas las solicitudes HTTP se dirigirán al generador de anonimato, el cual solicita la página y la envía de vuelta. El sitio web ve a un nodo de salida de la red del generador de anonimato como el origen de la solicitud, no al usuario. Mientras que la red del generador de anonimato se abstenga de mantener un registro, después del hecho nadie podrá determinar quién solicitó qué página.

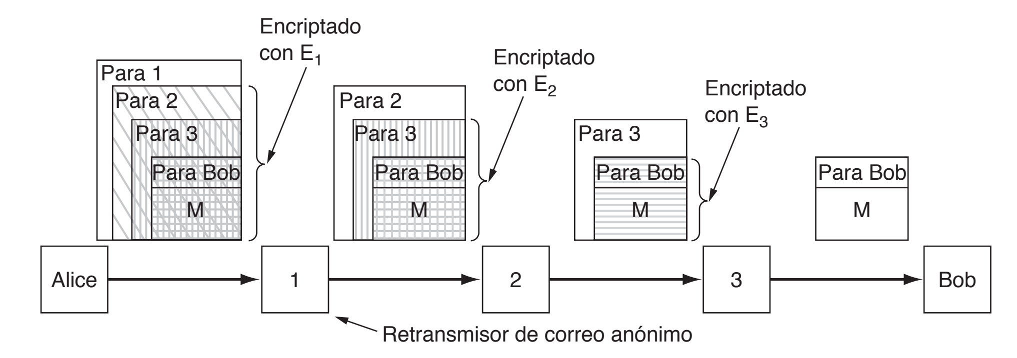


Figura 8-53. Forma en que Alice usa tres retransmisores de correo para enviar un mensaje a Bob.

8.10.2 Libertad de expresión

La privacidad se refiere a los individuos que desean restringir lo que otras personas ven en ellos. Un segundo problema social clave es la libertad de expresión, y su aspecto opuesto, la censura, que tiene que ver con el hecho de que los gobiernos desean restringir lo que los individuos pueden leer y publicar. Debido a que la web contiene millones y millones de páginas, se ha vuelto un paraíso de censura. Dependiendo de la naturaleza e ideología del régimen, entre el material prohibido podrían encontrarse los sitios web que contengan cualquier contenido de los siguientes tipos:

1. Material inapropiado para niños o adolescentes.
2. Odio dirigido a varios grupos religiosos, étnicos o sexuales, entre otros.
3. Información sobre democracia y valores democráticos.
4. Relatos de eventos históricos que contradigan la versión del gobierno.
5. Manuales para abrir candados, construir armas, encriptar mensajes, etcétera.

La respuesta común es prohibir los sitios "malos".

Algunas veces los resultados son inesperados. Por ejemplo, algunas bibliotecas públicas han instalado filtros web en sus computadoras de modo que sean aptas para los niños, para lo cual bloquean los sitios pornográficos. Los filtros vetan los sitios que se encuentran en sus listas negras y también verifican las páginas antes de desplegarlas para ver si contienen palabras obscenas. En una ocasión en el condado Loudoun, en Virginia, sucedió que el filtro bloqueó la búsqueda que un cliente realizó para encontrar información sobre el cáncer de seno porque el filtro vio la palabra "seno". Dicho usuario de la biblioteca demandó al condado Loudoun. Sin embargo, en Livermore, California, un padre demandó a la biblioteca pública por *no* haber instalado un filtro después de que sorprendió a su hijo de 12 años de edad viendo pornografía ahí. ¿Qué pueden hacer lo encargados de la biblioteca?

Mucha gente ha obviado el hecho de que la World Wide web es una red mundial. Cubre a todo el mundo. No todos los países están de acuerdo con lo que se debe permitir en la web. Por ejemplo, en noviembre de 2000, una corte de Francia ordenó a Yahoo!, una corporación de California, que bloqueara a sus usuarios franceses para que no pudieran ver las subastas de objetos de recuerdo nazis en el sitio web de Yahoo!, porque poseer tal material viola las leyes francesas. Yahoo! apeló en una corte de Estados Unidos, la cual le dio la razón, pero aún está lejos de resolverse el problema de dónde aplicar las leyes de quién.

Tan solo imagine. ¿Qué pasaría si alguna corte de Utah ordenara a Francia que bloqueara los sitios web relacionados con el vino porque no cumplen con las muy estrictas leyes de Utah sobre el alcohol? Suponga que China demandara que todos los sitios web que tienen que ver con la democracia fueran prohibidos porque no son del interés del Estado. ¿Se aplican las leyes iraníes sobre la religión a Suecia, que es más liberal? ¿Puede Arabia Saudita bloquear los sitios web que tienen que ver con los derechos de la mujer? Todo el problema es una auténtica caja de Pandora.

Un comentario relevante de John Gilmore es: "La Red interpreta la censura como una avería y encuentra una ruta alterna". Para una implementación concreta, considere el **servicio eternidad** (*ethernity service*) (Anderson, 1996). Su objetivo es asegurarse de que la información publicada no se pueda eliminar o reescribir, como era común en la antigua Unión Soviética durante el gobierno de Josef Stalin. Para utilizar el servicio eternidad, el usuario debe especificar cuánto tiempo se mantendrá el material, paga una cuota proporcional a su duración y tamaño, y lo envía. Después de eso, nadie puede eliminarlo o modificarlo, ni siquiera quien lo envió.

¿Cómo se puede implementar tal servicio? El modelo más sencillo es utilizar un sistema de igual a igual en el que los documentos almacenados se coloquen en docenas de servidores participantes, cada uno

de los cuales obtiene una parte de la cuota y, por lo tanto, un incentivo para unirse al sistema. Los servidores deben esparcirse a través de muchas jurisdicciones legales para obtener una máxima resistencia. Las listas de 10 servidores seleccionados al azar se podrían almacenar en forma segura en varios lugares, por lo que si algunos estuvieran en peligro, otros aún existirían. Una autoridad dispuesta a destruir el documento nunca estará segura de que ha encontrado todas las copias. El sistema también podría repararse a sí mismo; por ejemplo, si se sabe que se han destruido algunas copias, los sitios restantes podrían intentar encontrar nuevos depósitos para reemplazarlas.

El servicio eternidad fue la primera propuesta para un sistema resistente a la censura. Desde entonces se han propuesto otros sistemas y, en algunos casos, se han implementado. Asimismo, se han agregado algunas nuevas características, como encriptación, anonimato y tolerancia a fallas. Con frecuencia los archivos se dividen en múltiples fragmentos, los cuales se almacenan en muchos servidores. Algunos de estos sistemas son Freenet (Clarke y colaboradores, 2002), PASIS (Wylie y colaboradores, 2000) y Publius (Waldman y colaboradores, 2000). En Serjantov (2002) se reporta más trabajo.

En la actualidad, cada vez más países tratan de regular la exportación de valores intangibles, entre los que se encuentran comúnmente sitios web, software, documentos científicos, correo electrónico, servicios de ayuda telefónica, entre otros. Incluso en el Reino Unido, que tiene una tradición de siglos de libertad de expresión, ahora está considerando seriamente las leyes muy restrictivas, las cuales podrían, por ejemplo, definir las discusiones técnicas entre un profesor británico y su estudiante extranjero de doctorado, ambos ubicados en la Universidad de Cambridge, como exportación regulada que necesita una licencia del gobierno (Anderson, 2002). No es necesario decir que muchas personas consideran tales políticas como controversiales.

Esteganografía

En los países en donde abunda la censura, los disidentes con frecuencia tratan de utilizar la tecnología para evadirla. La criptografía permite el envío de mensajes secretos (aunque tal vez no en forma legal), pero si el gobierno piensa que Alice es una mala persona, el simple hecho de que ella se esté comunicando con Bob podría ponerlo a él también en esta categoría, pues los gobiernos represivos entienden el concepto de clausura transitiva, aun cuando tengan escasez de matemáticos. Los retransmisores de correo anónimos pueden ayudar, pero si están prohibidos en el hogar y los mensajes dirigidos a extranjeros requieren una licencia de exportación por parte del gobierno, no serán de mucha ayuda. Pero la web sí puede.

Las personas que desean comunicarse de manera secreta con frecuencia tratan de ocultar el hecho de que se está realizando la comunicación. La ciencia de ocultar mensajes se conoce como **esteganografía**, cuyo origen proviene de las palabras griegas correspondientes a “escritura encubierta”. De hecho, los antiguos griegos la utilizaron. Heródoto escribió sobre un general que rapó a un mensajero, tatuó un mensaje en el cuero cabelludo de éste y dejó que le creciera el cabello antes de enviarlo a realizar la entrega. En concepto, las técnicas modernas son iguales, sólo que tienen un mayor ancho de banda, una menor latencia y no requieren los servicios de un peluquero.

Como ejemplo, considere la figura 8-54(a). Esta fotografía, que tomó uno de los autores (AST) en Kenia, contiene tres cebras que miran hacia un árbol de acacia. La figura 8-54(b) parece ser la misma foto de las tres cebras y el árbol de acacia, pero tiene una atracción extra. Contiene todo el texto íntegro de cinco obras de Shakespeare: *Hamlet*, *El Rey Lear*, *Macbeth*, *El Mercader de Venecia* y *Julio César*. Juntas, estas obras tienen un tamaño aproximado a los 700 KB de texto.

¿Cómo funciona este canal esteganográfico? La imagen a color original es de 1024×768 píxeles. Cada píxel consiste en tres números de 8 bits, cada uno para la intensidad de rojo, verde y azul de ese píxel. El color del píxel se forma mediante la superposición lineal de los tres colores. El método de codificación

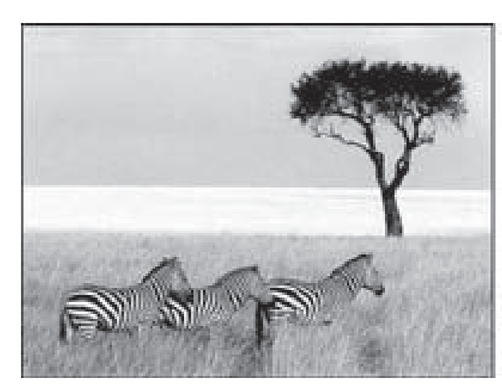
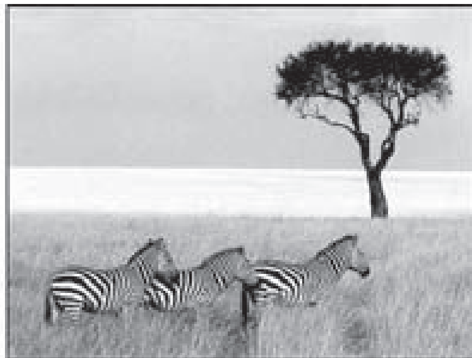



Figura 8-54. (a) Tres cebras y un árbol. (b) Tres cebras, un árbol y el texto completo de cinco obras de William Shakespeare.

esteganográfica utiliza como canal secreto el bit de menor orden de cada valor de color RGB. De esta manera, cada píxel tiene espacio para 3 bits de información secreta, uno en el valor de rojo, otro en el valor de verde y otro más en el de azul. En una imagen de este tamaño se pueden almacenar hasta 1024 × 768 × 3 bits o 294 912 bytes de información secreta.

Todo el texto de las cinco obras y una noticia corta suman 734 891 bytes. Este texto primero se comprimió a casi 274 KB mediante un algoritmo de compresión estándar. Después, la salida comprimida se encriptó mediante el uso de IDEA y se insertó en los bits de menor orden de cada valor de color. Como podemos ver (o más bien, no podemos), la información es por completo invisible. También es invisible en la versión a todo color de la foto. El ojo no puede distinguir con facilidad los colores de 21 bits de los de 24 bits.

El hecho de ver las dos imágenes en blanco y negro con una resolución baja no hace justicia al grado de poder de esta técnica. Para tener una mejor idea de cómo funciona la esteganografía, hemos preparado una demostración, que incluye la imagen de alta resolución a todo color de la figura 8-54(b) con las cinco obras incrustadas en ella. En el sitio web del libro se incluye la demostración, junto con las herramientas para insertar y extraer texto en imágenes.

Para utilizar la esteganografía para la comunicación secreta, los disidentes podrían crear un sitio web que contenga imágenes políticamente correctas, como fotografías de un gran líder, de deportes locales, de estrellas de películas y de televisión, etc. Por supuesto que las imágenes contendrán mensajes esteganográficos. Si los mensajes primero se comprimieran y después se encriptaran, incluso alguien que sospechara su presencia tendría mucha dificultad para distinguir dichos mensajes del ruido blanco. Cabe mencionar que las imágenes deben ser digitalizaciones recientes; el hecho de copiar una imagen de Internet y cambiar algunos de los bits produce una revelación involuntaria.

Las imágenes no son el único medio para los mensajes esteganográficos. Los archivos de audio también funcionan bien. La información oculta se puede transportar en una llamada de voz sobre IP al manipular los retardos de los paquetes, distorsionar el audio o incluso en los campos de encabezado de los paquetes (Lubacz y colaboradores, 2010). Incluso la distribución y el orden de las etiquetas de un archivo HTML pueden llevar información.

Aunque hemos examinado la esteganografía en el contexto de la libertad de expresión, tiene varios usos más. Uno de los más comunes es para que los propietarios de imágenes codifiquen mensajes secretos en ellas e indiquen de esa forma sus derechos de propiedad. Si una imagen de éstas es robada y se coloca en un sitio web, el propietario legal puede revelar el mensaje esteganográfico en la Corte para probar a quién pertenece esa imagen. Esta técnica se conoce como **marca de agua** (*watermarking*) y se describe en Piva y colaboradores (2002).

Para obtener mayor información sobre la esteganografía, consulte a Wayner (2008).

8.10.3 Derechos de autor

La privacidad y la censura son sólo dos áreas en las que la tecnología se encuentra con la política pública. La tercera es la ley de los derechos de autor. Éstos son el otorgamiento a los creadores de la **PI** (**Propiedad Intelectual**), incluyendo a los escritores, poetas, artistas, compositores, músicos, fotógrafos, cinematógrafos, coreógrafos, entre otros, del derecho exclusivo para explotar su PI durante cierto periodo, por lo general durante la vida del autor más 50 o 75 años en el caso de la propiedad corporativa. Después de que expiran los derechos de autor de algún trabajo, pasa a ser del dominio público y cualquiera puede utilizarlo o venderlo según le plazca. Por ejemplo, el Proyecto Gutenberg (*www.promo.net/pg*) ha colocado en web miles de trabajos del dominio público (por ejemplo, de Shakespeare, Twain y Dickens). En 1998, el Congreso de Estados Unidos extendió por 20 años más los derechos de autor en ese país por solicitud de Hollywood, que afirmó que si no se otorgaba una extensión, nadie crearía nada más. En contraste, las patentes sólo duran 20 años y las personas aún siguen inventando cosas.

Los derechos de autor dieron de qué hablar cuando Napster, un servicio de intercambio de música, tenía 50 millones de miembros. Aunque Napster realmente no copiaba la música, las Cortes sostuvieron que el hecho de que mantuviera una base de datos de quién tenía las canciones era infracción contributiva; es decir, Napster ayudaba a otras personas a infringir la ley. Si bien nadie alega que los derechos de autor son una mala idea (aunque muchas personas alegan que el término es demasiado largo, lo que favorece a las grandes corporaciones y no al público), la siguiente generación de compartición de música ya está provocando problemas mayores de ética.

Por ejemplo, considere una red de igual a igual en la que las personas comparten archivos legales (música del dominio público, videos caseros, pistas religiosas que no son secretos comerciales, etc.), y que algunos de los cuales tal vez tengan derechos de autor. Suponga que todos están en línea todo el día mediante ADSL o cable. Cada máquina tiene un índice de lo que hay en el disco duro, más una lista de otros miembros. Alguien que esté buscando un elemento específico puede elegir un miembro al azar y ver si éste tiene tal elemento. Si no lo tiene, puede verificar a todos los miembros que se encuentran en la lista de esa persona y, si no lo encuentra, a todos los miembros que se encuentren en las listas de dichos miembros, y así en lo sucesivo. Las computadoras son muy buenas para este tipo de trabajo. Cuando encuentra el elemento, el solicitante simplemente lo copia.

Si el trabajo tiene derechos de autor, es probable que el solicitante esté infringiendo la ley (aunque para las transferencias internacionales es importante la pregunta de cuál ley se aplica, ya que en algunos países es ilegal subir archivos pero no descargarlos). Pero, ¿qué sucede con el proveedor? ¿Es un crimen mantener la música por la que usted ha pagado y que ha descargado legalmente en su disco duro, en donde otros pueden encontrarla? Si usted tiene un gabinete sin candado en el país y un ladrón de PI entra con una computadora portátil y un digitalizador, copia un libro que tiene derechos de autor al disco duro de la computadora portátil y se va de manera furtiva, ¿es *usted* culpable por no proteger los derechos de autor de alguien más?

Pero hay más problemas relacionados con los derechos de autor. En la actualidad hay una gran batalla entre Hollywood y la industria de la computación. El primero quiere protección estricta de toda la propiedad intelectual, pero la segunda no desea ser el policía de Hollywood. En octubre de 1998, el Congreso aprobó la **DMCA** (**Ley de Derechos de Autor para el Milenio Digital**, del inglés *Digital Millennium Copyright Act*), que considera un crimen evadir cualquier mecanismo de protección presente en un trabajo con derechos de autor o indicar a otros cómo evadirlo. En la Unión Europea se ha establecido una legislación similar. Si bien casi nadie piensa que debería permitirse a los piratas del Lejano Oriente duplicaran los trabajos con derechos de autor, muchas personas piensan que la DMCA desplaza por completo el balance entre los intereses de los propietarios con derechos de autor y el interés público.

Por ejemplo, en septiembre de 2000, un consorcio de la industria de la música encargado de construir un sistema inquebrantable para vender música en línea patrocinó un concurso en el que invitaba a las

personas a que trataran de quebrantar el sistema (que es precisamente lo que debería hacerse con cualquier sistema de seguridad nuevo). Un equipo de investigadores de seguridad de diversas universidades, dirigidas por el profesor Edward Felten de Princeton, aceptó el desafío y quebrantó el sistema. Después, este equipo escribió un documento sobre sus descubrimientos y lo envió a la conferencia de seguridad USENIX, en donde se le sometió a evaluación por expertos y fue aceptado. Antes de que el documento se presentara, Felten recibió una carta de la Asociación Estadounidense de la Industria Discográfica (*Recording Industry Association of America*), la cual amenazaba con demandarlos apoyándose en la DMCA si publicaban el documento.

Su respuesta fue ir a juicio y pedir a una corte federal que reglamentara sobre si el hecho de publicar documentos científicos sobre investigación de seguridad era todavía legal. Temiendo que la corte fallara en contra de ellos, la industria retiró su amenaza y la corte desechó la demanda de Felten. Sin duda la industria estaba motivada por la debilidad de su caso: habían invitado a las personas a que trataran de quebrantar su sistema y después amenazaron con demandar a algunas de ellas por aceptar su propio desafío. Una vez retirada la amenaza, el documento se publicó (Craver y colaboradores, 2001). Sin duda es inminente una nueva confrontación.

Mientras tanto, la música y las películas piratas han alimentado el crecimiento masivo de las redes de igual a igual. Esto no tiene contentos a los propietarios de los derechos de autor, quienes han utilizado la DCMA para emprender acciones legales. Ahora hay sistemas automatizados que buscan en las redes de igual a igual y después emiten advertencias para los operadores de las redes y los usuarios sospechosos de infringir los derechos de autor. En Estados Unidos estas advertencias se conocen como **notificaciones de retirada de la DMCA**. Esta búsqueda es una carrera armamentista, ya que es difícil atrapar de manera confiable a los infractores de los derechos de autor. Incluso hasta su impresora podría ser considerada por error como culpable (Piatek y colaboradores, 2008).

Un asunto relacionado es la extensión de la **doctrina de uso razonable**, que ha sido establecida por fallos de la corte en varios países. Esta doctrina establece que los que compran un trabajo con derechos de autor tienen ciertos derechos limitados para copiar el trabajo, incluyendo el derecho de citar partes de él para propósitos científicos, utilizarlo como material de enseñanza en escuelas o colegios y, en algunos casos, realizar copias de respaldo para uso personal en caso de que el medio original falle. Las pruebas de lo que constituye un uso razonable incluyen (1) si el uso es comercial, (2) qué porcentaje se copia de todo el trabajo, y (3) el efecto del copiado en las ventas del trabajo. Puesto que la DMCA y las leyes similares dentro de la Unión Europea prohíben la evasión de los esquemas de protección contra copia, estas leyes también prohíben el uso razonable legal. En efecto, la DMCA elimina los derechos históricos de los usuarios para dar más poder a los vendedores de contenido. Es inevitable una confrontación mayor.

Otro desarrollo en los trabajos que eclipsa incluso a la DMCA en su desplazamiento del equilibrio entre los propietarios de los derechos de autor y los usuarios es la **computación confiable**, según la recomendación de organizaciones industriales tales como el **TCG** (**Grupo de Computación Confiable**, del inglés *Trusted Computing Group*), dirigido por empresas como Intel y Microsoft. La idea es proveer soporte para supervisar de manera cuidadosa el comportamiento de los usuarios de diversas formas (por ejemplo, al reproducir música pirata) a un nivel inferior al sistema operativo, para prohibir el comportamiento indeseable. Esto se logra mediante un pequeño chip, conocido como **TPM** (**Módulo de Plataforma Confiable**, del inglés *Trusted Platform Module*), el cual es difícil de falsificar. La mayoría de las PC que se venden en la actualidad vienen equipadas con un TPM. El sistema permite que el software escrito por los propietarios de contenido manipule a las PC en formas que los usuarios no pueden cambiar. Aquí surge la pregunta acerca de en quién se puede confiar en la computación confiable. Sin duda, no en el usuario. No es necesario decir que las consecuencias sociales de este esquema son inmensas. Es bueno que la industria finalmente esté poniendo atención a la seguridad, pero es lamentable que el enfoque sea en cumplir las leyes de derechos de autor en lugar de lidiar con los virus, crackers, intrusos y otros problemas de seguridad por los que la mayoría de la gente está preocupada.

En resumen, los legisladores y abogados estarán ocupados equilibrando los intereses económicos de los propietarios de derechos de autor con los intereses públicos en los años por venir. El ciberespacio no es muy diferente de la realidad: hay confrontamientos constantes entre un grupo y otro, lo que resulta en batallas por el poder, litigaciones y llegar (con suerte) en un momento dado a algún tipo de resolución, por lo menos hasta que aparezca una nueva tecnología inquietante.

8.11 RESUMEN

La criptografía es una herramienta que se puede utilizar para mantener confidencial la información y para asegurar tanto su integridad como su autenticidad. Todos los sistemas criptográficos modernos se basan en el principio de Kerckhoff de tener un algoritmo conocido públicamente y una clave secreta. Muchos algoritmos criptográficos utilizan transformaciones complejas que involucran sustituciones y permutaciones para transformar el texto plano en texto cifrado. No obstante, si la criptografía cuántica puede hacerse práctica, el uso de rellenos de una sola vez podría proporcionar criptosistemas realmente inquebrantables.

Los algoritmos criptográficos se pueden dividir en algoritmos de clave simétrica y algoritmos de clave pública. Los primeros alteran los bits en una serie de rondas parametrizadas por la clave para convertir el texto plano en texto cifrado. En la actualidad los algoritmos de clave simétrica más populares son AES (Rijndael) y el triple DES. Éstos se pueden utilizar en modo de libro de código electrónico, modo de encadenamiento de bloque de sistema cifrado, modo de sistema de cifrado de flujo, modo de contador, y varios más.

Los algoritmos de clave pública tienen la propiedad de que se utilizan diferentes claves para la encriptación y desencriptación; además, la clave de desencriptación no se puede obtener de la clave de encriptación. Gracias a estas propiedades es posible dar a conocer la clave pública. El principal algoritmo de clave pública es RSA, el cual deriva su fuerza del hecho de que es muy difícil factorizar números grandes.

Los documentos legales, comerciales y de otro tipo necesitan firmarse. Asimismo, se han diseñado varios esquemas para las firmas digitales, las cuales utilizan tanto algoritmos de clave simétrica como de clave pública. Por lo general, a los mensajes que se van a firmar se les aplica un hash mediante algoritmos como SHA-1, y después se firman los hashes en lugar de los mensajes originales.

La administración de claves públicas se puede realizar mediante el uso de certificados, los cuales son documentos que enlazan a un personaje principal con una clave pública. Los certificados son firmados por una autoridad de confianza o por alguien aprobado (recursivamente) por una autoridad de confianza. La raíz de la cadena se tiene que obtener por adelantado, pero los navegadores por lo general tienen integrados muchos certificados raíz.

Estas herramientas criptográficas se pueden utilizar para asegurar el tráfico de la Red. IPsec opera en la capa de red y encripta flujos de paquetes de host a host. Los *firewalls* pueden filtrar el tráfico entrante y saliente de una organización, a menudo con base en el protocolo y puerto utilizado. Las redes privadas virtuales pueden simular una vieja red de línea alquilada para proporcionar ciertas propiedades de seguridad deseables. Por último, las redes inalámbricas necesitan buena seguridad para evitar que todos lean los mensajes, y los protocolos como el 802.11i la proporcionan.

Cuando dos partes establecen una sesión, se deben autentificar entre sí y, si es necesario, establecer una clave de sesión compartida. Existen varios protocolos de autentificación, entre ellos algunos que utilizan un tercero confiable, Diffie-Hellman, Kerberos y la criptografía de clave pública.

La seguridad en el correo electrónico se puede alcanzar mediante una combinación de las técnicas que hemos estudiado en este capítulo. Por ejemplo, PGP comprime los mensajes, después los encripta con una clave secreta y envía la clave secreta encriptada con la clave pública del receptor. Además, también aplica un hash al mensaje y envía el hash firmado para verificar la integridad del mensaje.

La seguridad en la web también es un tema importante, empezando con la asignación de nombres segura. DNSsec proporciona una forma de evitar la falsificación del DNS. La mayoría de los sitios web de comercio electrónico utilizan SSL/TLS para establecer sesiones autentificadas seguras entre el cliente y el servidor. Se utilizan varias técnicas para lidiar con el código móvil, en especial las cajas de arena y la firma de código.

Internet genera muchas cuestiones en las que la tecnología interactúa de manera considerable con la política pública. Algunas de estas áreas incluyen la privacidad, la libertad de expresión y los derechos de autor.

PROBLEMAS

1. Quebrante el siguiente sistema de cifrado de sustitución monoalfabética. El texto plano, que consiste sólo de letras, es un pasaje de un poema de Lewis Carroll.

   kfd ktbd fzm eubd kfd pzyiom mztx ku kzyg ur bzha kfthcmur mftnm zhx mfudm zhx mdzythc pzq ur ezsszcdm zhx gthcm zhx pfa kfd mdz tm sutythc fuk zhx pfdkfdi ntcm fzld pthcm sok pztk z stk kfd uamkdim eitdx sdruid pd fzld uoi efzk rui mubd ur om zid ouk ur sidzkf zhx zyy ur om zid rzk hu foiia mztx kfd ezindhkdi kfda kfzhgdx ftb boef rui kfzk

2. Un sistema de cifrado afín es una versión de un sistema de cifrado de sustitución monoalfabética, en donde las letras de un alfabeto de tamaño m se asocian primero a los enteros en el rango de 0 a $m - 1$. Después, el entero que representa a cada letra de texto plano se transforma en un entero que representa a la letra de texto cifrado correspondiente. La función de encriptación para una sola letra es $E(x) = (ax + b) \bmod m$, en donde m es el tamaño del alfabeto, a y b son la clave del sistema de cifrado, y además son coprimos. Trudy descubre que Bob generó un texto cifrado mediante el uso de un sistema cifrado afín. Ella obtiene una copia del texto cifrado y averigua que la letra más frecuente del mismo es "R", y que la segunda letra más frecuente del texto cifrado es "K". Muestre cómo Trudy puede quebrantar el código y obtener el texto plano.

3. Quebrante el siguiente sistema de cifrado de transposición columnar. El texto plano proviene de un libro de texto de computadoras muy popular, por lo que "computadora" es una palabra probable. El texto plano consiste totalmente de letras (sin espacios). Por cuestión de legibilidad, el texto cifrado está dividido en bloques de cinco caracteres.

   aauan cvlre rurnn dltme aeepb ytust iceat npmey iicgo gorch srsoc
   nntii imiha oofpa gsivt tpsit lbolr otoex

4. Alice utilizó un sistema de cifrado por transposición para encriptar sus mensajes para Bob. Como seguridad adicional, encriptó la clave de cifrado por transposición mediante el uso de un sistema de cifrado de sustitución, y guardó el sistema de cifrado encriptado en su computadora. Trudy logró apropiarse de la clave de cifrado por transposición encriptada. ¿Puede descifrar los mensajes que Alice envió a Bob? ¿Por qué sí o por qué no?

5. Encuentre un relleno de una sola vez de 77 bits que genere el texto "Hello World" a partir del texto cifrado de la figura 8-4.

6. Usted es un espía y, por conveniencia, tiene una biblioteca con un número infinito de libros a su disposición. Su operador también tiene una biblioteca similar a su disposición. Usted ha acordado usar *El señor de los anillos* como un relleno de una sola vez. Explique cómo podría usar estos recursos para generar un relleno de una sola vez con una longitud infinita.

7. La criptografía cuántica requiere tener un arma de fotones que pueda disparar a solicitud un solo fotón que transporte 1 bit. En este problema, calcule cuántos fotones transporta un bit en un enlace de fibra de 250 Gbps. Suponga que la longitud de un fotón es igual a su longitud de onda, que para los fines de este problema, es 1 micra. La velocidad de la luz en la fibra es de 20 cm/nseg.

8. Si Trudy captura y regenera fotones cuando está en uso la criptografía cuántica, obtendrá algunos fotones erróneos y causará errores en el relleno de una sola vez de Bob. ¿Qué fracción de los bits de relleno de una sola vez de Bob serán erróneos en promedio?

9. Un principio fundamental de criptografía establece que todos los mensajes deben tener redundancia. Pero también sabemos que la redundancia ayuda a que un intruso sepa si una clave adivinada es correcta. Considere dos formas de redundancia. Primero, los n bits iniciales del texto plano contienen un patrón conocido. Segundo, los n bits finales del mensaje contienen un hash sobre el mensaje. Desde un punto de vista de seguridad, ¿son estos dos equivalentes? Analice su respuesta.

10. En la figura 8-6, se alternan las cajas P y S. Aunque este arreglo sea estéticamente placentero, ¿es más seguro que primero tener todas las cajas P y después todas las S? Explique su respuesta.

11. Diseñe un ataque a DES con base en el conocimiento de que el texto plano consiste exclusivamente en letras ASCII mayúsculas, más un espacio, coma, punto, punto y coma, retorno de carro y avance de línea. No se sabe nada sobre los bits de paridad del texto plano.

12. En el texto calculamos que una máquina para quebrantar sistemas de cifrado con un millón de procesadores que pueden analizar una clave en 1 nanosegundo podría tardar 10^{16} años para quebrantar la versión de 128 bits de AES. Ahora vamos a calcular cuánto tiempo se requerirá para que este periodo se reduzca a 1 año, en el transcurso del tiempo, desde luego. Para lograr este objetivo, necesitamos que las computadoras sean 10^{16} veces más rápidas. Si la ley de Moore (el poder de cómputo se duplica cada 18 meses) sigue vigente, ¿cuántos años se necesitarán para que una computadora paralela pueda reducir el tiempo para quebrantar el sistema de cifrado a un año?

13. AES soporta una clave de 256 bits. ¿Cuántas claves tiene AES-256? Vea si puede obtener algún número en física, química o astronomía con el mismo tamaño aproximado. Utilice Internet para buscar números grandes. Elabore una conclusión a partir de su investigación.

14. Suponga que un mensaje se ha encriptado mediante el uso de DES en modo de contador. Un bit de texto cifrado en el bloque C_i se transforma por accidente de un 0 a un 1 durante la transmisión. ¿Cuánto texto plano quedará ilegible como resultado?

15. Ahora considere de nuevo el encadenamiento de bloque de texto cifrado. En lugar de que un solo bit 0 se transforme en un bit 1, se inserta un bit 0 adicional en el flujo de texto cifrado después del bloque C_i. ¿Cuánto texto plano se distorsionará como resultado?

16. Compare el encadenamiento de bloque de sistema cifrado con el modo de retroalimentación de sistema de cifrado en términos del número de operaciones de encriptación necesarias para transmitir un archivo extenso. ¿Cuál es más eficiente y por cuánto?

17. Use el criptosistema de clave pública RSA, con $a = 1, b = 2 \cdots y = 25, z = 26$.

(a) Si $p = 5$ y $q = 13$, liste cinco valores legales para d.
(b) Si $p = 5, q = 31$ y $d = 37$, encuentre e.
(c) Mediante el uso de $p = 3, q = 11$ y $d = 9$, encuentre e y encripte "hola".

18. Alice y Bob usan la encriptación de clave pública RSA para comunicarse entre ellos. Trudy descubre que Alice y Bob compartieron uno de los primos usados para determinar el número n de sus pares de claves públicas. En otras palabras, Trudy descubrió que $n_a = p_a \times q$ y que $n_b = p_b \times q$. ¿Cómo puede usar Trudy esta información para quebrantar el código de Alice?

19. Considere el uso del modo de contador, como se muestra en la figura 8-15, pero con $IV = 0$. ¿Acaso el uso de 0 amenaza la seguridad del sistema de cifrado en general?

20. En la figura 8-20 podemos ver la forma en que Alice puede enviar a Bob un mensaje firmado. Si Trudy reemplaza P, Bob puede detectarlo. Pero, ¿qué sucede si Trudy reemplaza tanto P como la firma?

21. Las firmas digitales tienen una debilidad potencial debido a los usuarios flojos. En las transacciones de comercio electrónico podría suscribirse un contrato y el usuario podría solicitar la firma de su hash SHA-1. Si el usuario no verifica que el contrato y el hash correspondan, firmará de manera inadvertida un contrato diferente. Suponga que la mafia trata de explotar esta debilidad para ganar algo de dinero. Los mafiosos establecen un

sitio web de paga (por ejemplo, de pornografía, apuestas, etc.) y piden a los nuevos clientes un número de tarjeta de crédito. Después envían al cliente un contrato en el que estipulan que éste desea utilizar su servicio y pagar con tarjeta de crédito, y le piden que lo firme a sabiendas de que la mayoría de los clientes simplemente firmarán sin verificar que el contrato y el hash correspondan. Muestre la forma en que la mafia puede comprar diamantes de un joyero legítimo de Internet y cargarlos a clientes inocentes.

22. Una clase de matemáticas tiene 25 estudiantes. ¿Cuál es la probabilidad de que todos los estudiantes hayan nacido en la primera mitad del año (entre enero 1 y junio 30)? ¿Cuál es la probabilidad de que por lo menos dos estudiantes tengan la misma fecha de nacimiento? Suponga que nadie nació en año bisiesto, por lo que hay 181 posibles fechas de nacimiento.

23. Después de que Ellen confesó a Marilyn que la engañó en el asunto de Tom, Marilyn decidió evitar este problema, para lo cual piensa dictar el contenido de los mensajes futuros en una máquina de dictado y dárselos a su nueva secretaria para que los teclee. A continuación, Marilyn planea examinar los mensajes en su terminal después de que la secretaria los haya tecleado para asegurarse de que contengan sus palabras exactas. ¿Puede aún la nueva secretaria utilizar el ataque de cumpleaños para falsificar un mensaje?, y de ser así, ¿cómo? *Sugerencia:* Sí puede hacerlo.

24. Considere el intento fallido de Alice para obtener la clave pública de Bob en la figura 8-23. Suponga que Bob y Alice ya comparten una clave secreta, pero Alice aún quiere la clave pública de Bob. ¿Hay ahora una forma para obtenerla de manera segura? De ser así, ¿cómo?

25. Alice se quiere comunicar con Bob mediante el uso de la criptografía de clave pública. Ella establece una conexión con alguien que espera sea Bob. Le pide su clave pública y él se la envía en texto plano, junto con un certificado X.509 firmado por la raíz CA. Alice ya tiene la clave pública de la raíz CA. ¿Qué pasos debe realizar Alice para verificar que está hablando con Bob? Suponga que a Bob no le importa con quién está hablando (por ejemplo, Bob es algún tipo de servicio público).

26. Suponga que un sistema utiliza PKI con base en una jerarquía con estructura de árbol de autoridades CA. Alice desea comunicarse con Bob y recibe un certificado de Bob firmado por una CA *X* después de establecer un canal de comunicación con Bob. Suponga que Alice nunca ha escuchado sobre *X*. ¿Qué pasos debe realizar Alice para verificar que está hablando con Bob?

27. ¿Es posible utilizar en modo de transporte IPsec con AH si alguna de las máquinas está detrás de una caja NAT? Explique su respuesta.

28. Alice desea enviar un mensaje a Bob mediante hashes SHA-1. Ella consulta con usted en relación con el algoritmo de firma apropiado que debe usar. ¿Qué le sugeriría?

29. Dé una razón por la cual se configuraría un firewall para inspeccionar el tráfico entrante. Dé una razón por la cual se configuraría para inspeccionar tráfico saliente. ¿Cree que las inspecciones vayan a tener éxito?

30. Suponga que una organización utiliza VPN para conectar de manera segura sus sitios a Internet. Jim, un usuario de la organización, usa la VPN para comunicarse con Mary, su jefa. Describa un tipo de comunicación entre Jim y Mary que no requiera el uso de encriptación o cualquier otro mecanismo de seguridad, y otro tipo de comunicación que requiera de encriptación o de otro mecanismo de seguridad. Explique su respuesta.

31. Cambie ligeramente un mensaje del protocolo de la figura 8-34 para hacerlo resistente al ataque de reflexión. Explique por qué funcionaría su modificación.

32. El intercambio de claves de Diffie-Hellman se utiliza para establecer una clave secreta entre Alice y Bob. Alice envía a Bob (227, 5, 82). Bob responde con (125). El número secreto x de Alice es 12, y el número secreto y de Bob es 3. Muestre cómo calculan Alice y Bob la clave secreta.

33. Dos usuarios pueden establecer una clave secreta compartida mediante el algoritmo de Diffie-Hellman, incluso aunque nunca se hayan conocido, no compartan secretos ni tengan certificados.

(a) Explique de qué forma es susceptible este algoritmo a un ataque de intermediario.
(b) ¿Cómo cambiaría esta susceptibilidad si n o g fueran secretos?

34. En el protocolo de la figura 8-39, ¿por qué *A* se envía en texto plano junto con la clave de sesión encriptada?

35. En el protocolo Needham-Schroeder, Alice genera dos desafíos, R_A y R_{A2}. Esto parece excesivo. ¿No habría bastado con uno solo?

36. Suponga que una organización utiliza Kerberos para la autentificación. En términos de seguridad y disponibilidad del servicio, ¿cuál es el efecto si AS o TGS se desactivan?

37. Alice utiliza el protocolo de autentificación de clave pública de la figura 8-43 para autentificar la comunicación con Bob. Sin embargo, al enviar el mensaje 7 Alice olvidó encriptar R_B. Ahora Trudy conoce el valor de R_B. ¿Necesitan Alice y Bob repetir el procedimiento de autentificación con nuevos parámetros para asegurar una comunicación segura? Explique su respuesta.

38. En el protocolo de autentificación de clave pública de la figura 8-43, en el mensaje 7, R_B se encripta con K_S. ¿Es necesaria esta encriptación, o podría haber sido adecuado regresarla en texto plano? Explique su respuesta.

39. Las terminales de punto de venta que utilizan tarjetas con banda magnética y códigos NIP tienen una falla fatal: un comerciante malicioso puede modificar su lector de tarjetas para capturar y almacenar toda la información de la tarjeta, así como el código NIP, con el fin de realizar posteriormente transacciones adicionales (falsas). La siguiente generación de terminales de punto de ventas utilizará tarjetas con una CPU completa, teclado y una pequeña pantalla en la tarjeta. Diseñe un protocolo para este sistema que los comerciantes maliciosos no puedan quebrantar.

40. ¿Es posible multidifundir un mensaje PGP? ¿Qué restricciones se aplicarían?

41. Suponiendo que todos en Internet utilizaron PGP, ¿podría un mensaje PGP enviarse a una dirección de Internet arbitraria y que todos los interesados lo decodificaran de manera correcta? Explique su respuesta.

42. El ataque mostrado en la figura 8-47 omite un paso. Éste no es necesario para que la falsificación funcione, pero si se incluye se podría reducir la sospecha potencial después del hecho. ¿Cuál es el paso faltante?

43. El protocolo de transporte de datos SSL involucra dos nonces además de una clave premaestra. ¿Qué valor (si es que lo hay) tiene el uso de los nonces?

44. Considere una imagen de 2048×512 píxeles. Usted quiere encriptar un archivo con un tamaño de 2.5 MB. ¿Qué fracción del archivo puede encriptar en esta imagen? ¿Qué fracción podría encriptar si comprimiera el archivo a una cuarta parte de su tamaño original? Muestre sus cálculos.

45. La imagen de la figura 8-54(b) contiene el texto ASCII de cinco obras de Shakespeare. ¿Sería posible ocultar música entre las cebras en lugar de ocultar texto? De ser así, ¿cómo funcionaría y cuánto podría ocultar en esta imagen? Si no es posible, explique por qué.

46. Usted recibe un archivo de texto con un tamaño de 60 MB, el cual se debe encriptar mediante el uso de esteganografía en los bits de menor orden de cada color en un archivo de imagen. ¿Qué tamaño de imagen se requeriría para encriptar todo el archivo? ¿Qué tamaño se necesitaría si el archivo se comprimiera primero a una tercera parte de su tamaño original? Muestre su respuesta en píxeles, junto con sus cálculos. Suponga que las imágenes tienen una proporción de aspecto de 3:2; por ejemplo, de $3\,000 \times 2\,000$ píxeles.

47. Alice usaba mucho un retransmisor de correo anónimo de tipo 1. Ella podía publicar muchos mensajes en su grupo de noticias favorito, *alt.fanclub.alice*, y todos sabían que tales mensajes provenían de Alice porque todos llevaban el mismo seudónimo. Suponiendo que el retransmisor de correo funcionaba de manera correcta, Trudy no podía suplantar a Alice. Después de que los retransmisores de correo de tipo 1 desaparecieron, Alice cambió a un retransmisor de correo cypherpunk e inició un nuevo hilo en su grupo de noticias. Diseñe una manera para que Alice evite que Trudy la suplante y publique nuevos mensajes en el grupo de noticias.

48. Busque en Internet un caso interesante que involucre la privacidad y escriba un informe de una página sobre él.

49. Busque en Internet algún caso de una corte que involucre los derechos reservados contra el uso razonable y escriba un informe de una página en el que resuma lo que haya encontrado.

50. Escriba un programa que encripte su entrada al aplicar a ésta un OR exclusivo con un flujo de claves. Busque o escriba lo mejor que pueda un generador de números aleatorios para generar el flujo de claves. El programa debe actuar como un filtro; debe tomar texto plano como entrada estándar para producir texto cifrado como salida estándar (y viceversa). El programa debe tomar un parámetro, la clave que alimenta al generador de números aleatorios.

51. Escriba un procedimiento que calcule el hash SHA-1 de un bloque de datos. El procedimiento debe tener dos parámetros: un apuntador al búfer de entrada y otro a un búfer de salida de 20 bytes. Para ver la especificación exacta de SHA-1, busque en Internet FIPS 180-1, que es la especificación completa.

52. Escriba una función que acepte un flujo de caracteres ASCII y encripte esta entrada mediante un sistema de cifrado de sustitución con el modo de encadenamiento de bloque de sistema de cifrado. El tamaño del bloque debe ser de 8 bytes. El programa debe tomar texto plano de la entrada estándar e imprimir el texto cifrado en la salida estándar. Para este problema puede seleccionar cualquier sistema razonable para determinar si se llegó al final de la entrada, y/o cuando haya que aplicar relleno para completar el bloque. Puede seleccionar cualquier formato de salida, siempre y cuando no sea ambiguo. El programa debe recibir dos parámetros:

(a) Un apuntador al vector de inicialización; y

(b) un número k que represente el desplazamiento del sistema de cifrado de sustitución, de tal forma que cada carácter ASCII se encripte mediante el k-ésimo carácter delante de él en el alfabeto. Por ejemplo, si $x = 3$, entonces A se codifica mediante D, B se codifica mediante E, etc. Haga suposiciones razonables con respecto a llegar al último carácter en el conjunto ASCII. Asegúrese de documentar con claridad en su código las suposiciones que haga sobre la entrada y el algoritmo de encriptación.

53. El propósito de este problema es que usted obtenga una mejor comprensión en cuanto a los mecanismos de RSA. Escriba una función que reciba como parámetros los primos p y q, que calcule las claves RSA pública y privada mediante estos parámetros, y las salidas n, z, d y e como impresiones en la salida estándar. Esta función también debe aceptar un flujo de caracteres ASCII y encriptar esta entrada mediante las claves RSA calculadas. El programa debe tomar texto plano de la entrada estándar e imprimir el texto cifrado en la salida estándar. La encriptación se debe realizar a nivel de carácter; es decir, hay que tomar cada carácter en la entrada y encriptarlo de manera independiente a los demás caracteres en la entrada. Para este problema, usted puede seleccionar cualquier sistema razonable para determinar si se llegó al final de la entrada. Puede seleccionar cualquier formato de salida, siempre y cuando no sea ambiguo. Asegúrese de documentar con claridad en su código las suposiciones que haga sobre la entrada y el algoritmo de encriptación.

9

LISTA DE LECTURAS Y BIBLIOGRAFÍA

Hemos terminado nuestro estudio de redes de computadoras, pero éste es sólo el comienzo. Por falta de espacio, no tratamos muchos temas interesantes con el detalle que se merecen y omitimos otros. Para beneficio de los lectores que desean continuar sus estudios de redes de computadoras, en este capítulo ofrecemos algunas sugerencias de lecturas adicionales y una bibliografía.

9.1 SUGERENCIAS DE LECTURAS ADICIONALES

Hay una extensa literatura sobre todos los aspectos de redes de computadoras. Dos revistas que publican artículos sobre esta área son *IEEE/ACM Transactions on Networking* y *IEEE Journal on Selected Areas in Communications*.

Las publicaciones periódicas del Grupo de Interés Especial sobre Comunicaciones de Datos de la ACM (SIGCOMM) y del Grupo de Interés Especial sobre Movilidad de los Sistemas, Usuarios, Datos y Computación (SIGMOBILE) publican muchos artículos de interés, en especial sobre temas emergentes. Éstas son *Computer Communication Review* y *Mobile Computing and Communications Review*.

El IEEE también publica tres revistas —*IEEE Internet Computing*, *IEEE Network Magazine* e *IEEE Communications Magazine*— que contienen encuestas, tutoriales y casos de estudios sobre redes. Las dos primeras se enfocan principalmente en la arquitectura, los estándares y el software, mientras que la última se orienta a la tecnología de comunicaciones (fibra óptica, satélites, etcétera).

Hay varias conferencias anuales o semestrales que captan numerosos artículos sobre redes. En especial, busque la conferencia *SIGCOMM*, el *NSDI* (Simposio sobre Diseño e Implementación de Sistemas en Red), *MobiSys* (Conferencia sobre Sistemas, Aplicaciones y Servicios

Móviles), *SOSP* (Simposio sobre Principios de los Sistemas Operativos) y *OSDI* (Simposio sobre Diseño e Implementación de Sistemas Operativos).

A continuación presentamos una lista de sugerencias de lectura adicional, relacionadas con los capítulos de este libro. Muchas de las sugerencias son capítulos o libros completos, con algunos tutoriales y estudios. Las referencias completas están en la sección 9.2.

9.1.1 Introducción y obras generales

Comer, *The Internet Book*, 4a. edición.

Cualquiera que desee una introducción fácil a Internet debe consultar este libro. Comer describe la historia, el crecimiento, la tecnología, los protocolos y servicios de Internet en términos que los principiantes pueden entender, pero abarca tanta información que el libro también es de interés para lectores más técnicos.

Computer Communication Review, edición del 25vo. aniversario, enero de 1995.

Para un análisis de primera mano sobre la forma en que se desarrolló Internet, esta edición especial reúne los artículos importantes hasta 1995. Se incluyen artículos que muestran el desarrollo de TCP, la multidifusión, el DNS, Ethernet y la arquitectura en general.

Crovella y Krishnamurthy, *Internet Measurement*.

¿Cómo sabemos qué tan bien funciona Internet? Esta pregunta no es de fácil respuesta, ya que no hay nadie a cargo de Internet. Este libro describe las técnicas que se han desarrollado para medir la operación de Internet, desde la infraestructura de la Red hasta las aplicaciones.

IEEE Internet Computing, enero-febrero de 2000.

La primera edición de *IEEE Internet Computing* en el nuevo milenio hizo exactamente lo que usted esperaría: preguntar a la gente qué ayudó a crear Internet en el milenio anterior y hacia dónde piensa que irá en el siguiente. Los expertos son Paul Baran, Lawrence Roberts, Leonard Kleinrock, Stephen Crocker, Danny Cohen, Bob Metcalfe, Bill Gates, Bill Joy, entre otros. Vea qué tan acertadas fueron sus predicciones después de más de una década.

Kipnis, "Beating the System: Abuses of the Standards Adoption Process".

Los comités de estándares tratan de ser justos y neutrales respecto a proveedores en cuanto a su trabajo pero, por desgracia, hay compañías que tratan de abusar del sistema. Por ejemplo, con frecuencia sucede que una compañía ayuda al desarrollo de un estándar y, una vez que éste se aprueba, anuncia que dicho estándar se basa en una patente que es de su propiedad, y que sólo dará licencias a compañías de su agrado, a precios que sólo ella determina. Si desea echar un vistazo al lado oscuro de la estandarización, este artículo es un excelente inicio.

Hafner y Lyon, *Where Wizards Stay Up Late*
Naughton, *A Brief History of the Future*

¿Quién inventó Internet, de todas formas? Muchas personas han reclamado el crédito. Y con justa razón, ya que muchas participaron en ello, en distintas formas. Uno de ellos fue Paul Baran, quien escribió un informe para describir la conmutación de paquetes; también estuvieron las personas de varias universidades que diseñaron la arquitectura de ARPANET, la gente de BBN que programó los primeros IMP, estuvieron Bob Kahn y Vint Cerf, quienes inventaron el TCP/IP, y muchos más. Estos libros cuentan la historia de Internet, por lo menos hasta el año 2000, y están repletos de anécdotas.

9.1.2 La capa física

Bellamy, *Digital Telephony*, 3a. edición.

Para una retrospectiva de la red telefónica, que también es una red importante, este libro autorizado contiene todo lo que siempre quiso saber sobre el sistema telefónico, entre muchas cosas más. Los capítulos sobre transmisión y multiplexado, conmutación digital, fibra óptica, telefonía móvil y DSL son muy interesantes.

Hu y Li, "Satellite-Based Internet: A Tutorial".

El acceso a Internet vía satélite es distinto al uso de líneas terrestres. No sólo tenemos la cuestión del retardo, sino que también el enrutamiento y la conmutación son distintos. En este artículo los autores examinan las cuestiones relacionadas con el uso de satélites para acceder a Internet.

Joel, "Telecommunications and the IEEE Communications Society"

Para una historia compacta pero sorprendentemente exhaustiva de las telecomunicaciones, que empieza con el telégrafo y termina con el 802.11, este artículo es el lugar adecuado. También trata sobre la radio, los teléfonos, la conmutación analógica y digital, los cables submarinos, la transmisión digital, la difusión de televisión, los satélites, la TV por cable, las comunicaciones ópticas, los teléfonos móviles, la conmutación de paquetes, ARPANET e Internet.

Palais, *Fiber Optic Communication*, 5a. edición.

Los libros sobre la tecnología de fibra óptica tienden a estar orientados al especialista, pero éste es más accesible que la mayoría. Habla sobre las guías de ondas, las fuentes de luz, los detectores de luz, los acopladores, la modulación, el ruido y muchos otros temas más.

Su, *The UMTS Air Interface in RF Engineering*.

Este libro proporciona una visión detallada de uno de los principales sistemas celulares 3G. Se enfoca en la interfaz aérea, o en los protocolos inalámbricos que se utilizan entre los móviles y la infraestructura de red.

Want, *RFID Explained*.

El libro de Want es una primicia de fácil lectura sobre cómo funciona la inusual tecnología de la capa física RFID. Cubre todos los aspectos sobre RFID, incluyendo sus aplicaciones potenciales. También incorpora algunos ejemplos reales sobre implementaciones de RFID y la experiencia obtenida a partir de éstas.

9.1.3 La capa de enlace de datos

Kasim, *Delivering Carrier Ethernet*.

En la actualidad, Ethernet no es sólo una tecnología de área local. La nueva tendencia es usar Ethernet como enlace de larga distancia para la Ethernet con grado de portadora. Este libro reúne ensayos para cubrir el tema a profundidad.

Lin y Costello, *Error Control Coding*, 2a. edición.

Los códigos para detectar y corregir errores son fundamentales para las redes de computadoras confiables. Este popular libro de texto explica algunos de los códigos más importantes, desde los simples códigos Hamming lineales hasta los códigos de verificación de paridad de baja densidad más complejos. Intenta hacerlo con el mínimo de álgebra necesario, pero de todas formas es mucho.

Stallings, *Data and Computer Communications*, 9a. edición.

La segunda parte dos trata sobre la transmisión de datos digitales y una variedad de enlaces, incluyendo la detección de errores, el control de los mismos mediante retransmisiones y el control de flujo.

9.1.4 La subcapa de control de acceso al medio

Andrews y colaboradores, *Fundamentals of WiMAX*.

Este extenso libro ofrece un análisis definitivo de la tecnología WiMAX, desde la idea de la red inalámbrica de banda ancha, pasando por las técnicas inalámbricas que utilizan OFDM y varias antenas, hasta llegar al sistema multiacceso. Su estilo de tutorial ofrece el análisis más accesible que pueda encontrar para este complejo material.

Gast, *802.11 Wireless Networks*, 2a. edición.

Si desea una introducción legible a la tecnología y a los protocolos del 802.11, éste es un buen libro para empezar. Primero cubre la subcapa MAC y después introduce material sobre las distintas capas físicas, y sobre la seguridad. Sin embargo, la segunda edición no está muy actualizada y no habla mucho sobre el 802.11n.

Perlman, *Interconnections*, 2ª edición

Si desea un análisis autoritario pero entretenido de los puentes, enrutadores y el enrutamiento en general, el libro de Perlman es el adecuado. La autora diseñó los algoritmos que se utilizan en el puente del árbol de expansión del IEEE 802 y es una de las principales autoridades en el mundo con relación a diversos aspectos de las redes.

9.1.5 La capa de red

Comer, *Internetworking with TCP/IP*, Vol. 1, 5a. edición.

Comer ha escrito la obra definitiva sobre la suite de protocolos TCP/IP, ahora en su quinta edición. Gran parte de la primera mitad trata sobre el IP y los protocolos relacionados en la capa de red. Los demás capítulos tratan principalmente sobre las capas superiores, por lo que también vale la pena leerlos.

Grayson y colaboradores, *IP Design for Mobile Networks*.

Las redes de telefonía tradicionales e Internet están en curso de colisión; las redes de teléfonos móviles se están implementando con IP en el interior. Este libro le dice cómo diseñar una red mediante los protocolos IP para soportar el servicio telefónico móvil.

Huitema, *Routing in the Internet*, 2a. edición.

Si desea un análisis detallado de los protocolos de enrutamiento, éste es un libro muy bueno. Tanto los algoritmos pronunciables (por ejemplo, RIP y CIDR) como los impronunciables (como OSPF, IGRP y BGP) se examinan con gran detalle. No se cubren los nuevos acontecimientos, ya que es un libro antiguo, pero lo que se cubre se explica muy bien.

Koodli y Perkins, *Mobile Inter-networking with IPv6*.

Aquí se presentan dos importantes desarrollos de la capa de red en un solo volumen: IPv6 e IP móvil. Ambos temas se cubren bien; además, Perkins fue una de las fuerzas impulsoras detrás de IP móvil.

Nucci y Papagiannaki, *Design, Measurement and Management of Large-Scale IP Networks*.

Vimos mucho sobre la forma en que funcionan las redes, pero no cómo diseñar, implementar y administrar una desde el punto de vista de un ISP. Este libro llena ese vacío; busca los métodos modernos para la ingeniería de tráfico y cómo los ISP proveen servicios mediante el uso de redes.

Perlman, *Interconnections*, 2a. edición.

En los capítulos 12 al 15, Perlman describe muchas de las cuestiones involucradas en el diseño de algoritmos de enrutamiento unidifusión y multidifusión, tanto para redes de área amplia como para redes compuestas a su vez por redes LAN. Pero hasta ahora, la mejor parte del libro es el capítulo 18, en donde la autora revela sus muchos años de experiencia con los protocolos de red en un capítulo informativo y divertido. Es una lectura obligada para los diseñadores de protocolos.

Stevens, *TCP/IP Illustrated*, vol. 1.

Los capítulos 3 al 10 ofrecen un análisis exhaustivo de IP y los protocolos relacionados (ARP, RARP e ICMP), ilustrados mediante ejemplos.

Varghese, *Network Algorithmics*.

Hemos pasado mucho tiempo hablando sobre la forma en que los enrutadores y otros elementos de red interactúan entre sí. Este libro es distinto: trata sobre cómo están realmente diseñados los enrutadores para reenviar paquetes a velocidades prodigiosas. Para obtener la información veraz sobre ésta y otras preguntas relacionadas, éste es el libro adecuado. El autor es una autoridad sobre los algoritmos inteligentes que se utilizan en la práctica para implementar elementos de red de alta velocidad en software y hardware.

9.1.6 La capa de transporte

Comer, *Internetworking with TCP/IP*, vol. 1, 5a. edición.

Como dijimos antes, Comer escribió la obra definitiva sobre la suite de protocolos TCP/IP. La segunda mitad del libro trata sobre UDP y TCP.

Farrell y Cahill, *Delay- and Disruption- Tolerant Networking*.

Este libro corto es el que debe leer para un análisis detallado sobre la arquitectura, protocolos y aplicaciones de las “redes desafiadas” que deben operar bajo conexiones severas de conectividad. Los autores han participado en el desarrollo de redes DTN en el Grupo de Investigación de DTN de la IETF.

Stevens, *TCP/IP Illustrated,* vol. 1.

Los capítulos 17 a 24 ofrecen un análisis exhaustivo de TCP, ilustrado mediante ejemplos.

9.1.7 La capa de aplicación

Berners-Lee y colaboradores, “The World Wide Web”

Viaje al pasado para obtener una perspectiva sobre la web y el futuro que le depara, a través de la persona que la inventó junto con algunos de sus colegas en el CERN. Este artículo se enfoca en la arquitectura de la Web, los URL, HTTP y HTML, así como las direcciones futuras, y la compara con otros sistemas de información distribuidos.

Held, *A Practical Guide to Content Delivery Networks*, 2a. edición.

Este libro ofrece una exposición realista sobre la forma en que funcionan las redes CDN, con un énfasis en las consideraciones prácticas sobre el diseño y la operación de una CDN que se desempeña bien.

Hunter y colaboradores, *Beginning XML*, 4a. edición.

Hay muchos, pero muchos libros sobre HTML, XML y los servicios web. Este libro de 1000 páginas cubre la mayoría de lo que quizás usted quiera saber. Explica no sólo cómo escribir XML y XHTML, sino también cómo desarrollar servicios web que produzcan y manipulen XML mediante Ajax, SOAP y otras técnicas de uso común en la práctica.

Krishnamurthy y Rexford, *Web Protocols and Practice*.

Sería difícil encontrar un libro más detallado que éste sobre todos los aspectos de web. Trata acerca de los clientes, servidores, proxies y el uso de caché, como podría esperar. Pero también hay capítulos sobre el tráfico web y las mediciones, así como capítulos sobre la investigación actual y las mejoras de web.

Simpson, *Video Over IP*, 2a. edición.

El autor presenta un amplio análisis sobre la forma en que se puede usar la tecnología IP para transmitir video a través de redes, tanto en Internet como en redes privadas diseñadas para transportar video. Lo interesante es que este libro está orientado para el profesional de video que desea aprender sobre las redes, en vez de lo contrario.

Wittenburg, *Understanding Voice Over IP Technology*.

Este libro trata acerca de la forma en que funciona la tecnología de voz sobre IP, desde el transporte de los datos de audio mediante los protocolos IP y los problemas de calidad del servicio, hasta SIP y la suite H.323 de protocolos. Es necesariamente detallado debido al material, pero a la vez accesible y está dividido en unidades digeribles.

9.1.8 Seguridad en redes

Anderson, *Security Engineering*, 2a. edición.

Aquí se presenta una maravillosa mezcla de técnicas de seguridad expresadas en un análisis sobre la forma en que las personas las utilizan (y hacen mal uso de ellas). Es más técnico que *Secrets and Lies* pero menos técnico que *Network Security* (vea a continuación). Después de una introducción a las técnicas básicas de seguridad, se dedican capítulos enteros a varias aplicaciones, incluyendo las bancarias, de comando y control nuclear, impresión de seguridad, biométrica, seguridad física, armamento electrónico, seguridad de telecomunicaciones, comercio electrónico y protección de los derechos de autor.

Ferguson y colaboradores, *Cryptography Engineering*

Muchos libros le indican cómo funcionan los populares algoritmos criptográficos. Este libro le dice cómo usar la criptografía —por qué los protocolos criptográficos se diseñan de la manera en que lo hacen actualmente y cómo reunirlos en un sistema que cumpla con nuestras metas de seguridad—. Es un libro bastante compacto que constituye una lectura obligada para cualquiera que diseñe sistemas que dependan de la criptografía.

Fridrich, *Steganography in Digital Media*.

La esteganografía se remonta a la antigua Grecia, en donde la cera se derretía de tablas en blanco para plasmar mensajes secretos en la madera subyacente antes de volver a aplicar la cera. En la actualidad, los videos, el audio y demás contenido en Internet proveen distintas portadoras para mensajes secretos. En este libro se explican varias técnicas modernas para ocultar y encontrar información en imágenes.

Kaufman y colaboradores, *Network Security*, 2a. edición.

Este libro indispensable e ingenioso es el primer lugar para buscar información técnica sobre los algoritmos y protocolos de seguridad de redes. Aquí se explican los algoritmos y protocolos de clave secreta y pública, los hashes de mensajes, la autentificación, Kerberos, PKI, Ipsec, SSL/TLS y la seguridad del correo electrónico con sumo cuidado y con una extensión considerable, con muchos ejemplos. El capítulo 26 sobre el folclore de la seguridad es una verdadera joya. En la seguridad, la clave está en los detalles. Cualquiera que planee diseñar un sistema de seguridad que realmente se vaya a utilizar, aprenderá mucho del consejo práctico incluido en este capítulo.

Schneier, *Secrets and Lies*.

Si leyó *Cryptography Engineering* de principio a fin, sabrá todo lo necesario sobre los algoritmos criptográficos. Si después lee *Secrets and Lies* de principio a fin (lo cual podrá hacer en mucho menos tiempo), descubrirá que los algoritmos criptográficos no son todo. La mayoría de las debilidades de seguridad no se deben a algoritmos fallidos ni a claves demasiado cortas, sino a fallas en el entorno de seguridad. Si desea un análisis no técnico y fascinante sobre la seguridad en las computadoras en el sentido más amplio, este libro es una lectura excelente.

Skoudis y Liston, *Counter Hack Reloaded*, 2ª edición

La mejor forma de detener a un hacker es pensar como uno. Este libro le muestra cómo ven los hackers una red, y argumenta que la seguridad debe ser una función de todo el diseño de la red, no una idea de último momento basada en una tecnología específica. Cubre casi todos los ataques comunes, incluyendo los tipos de "ingeniería social" que se aprovechan de los usuarios que no siempre están familiarizados con las medidas de seguridad computacional.

9.2 BIBLIOGRAFÍA

ABRAMSON, N. "Internet Access Using VSATs", *IEEE Commun. Magazine*, vol. 38, págs. 60-68, julio de 2000.

AHMADI, S. "An Overview of Next-Generation Mobile WiMAX Technology", *IEEE Commun. Magazine*, vol. 47, págs. 84-88, junio de 2009.

ALLMAN, M. y PAXSON, V. "On Estimating End-to-End Network Path Properties", *Proc. SIGCOMM '99 Conf.*, ACM, págs. 263-274, 1999.

ANDERSON, C. *The Long Tail: Why the Future of Business is Selling Less of More,* edición actualizada y revisada, Nueva York, Hyperion, 2008a.

ANDERSON, R. J. *Security Engineering: A Guide to Building Dependable Distributed Systems*, 2a. ed., Nueva York, John Wiley & Sons, 2008b.

————"Free Speech Online and Offline", *IEEE Computer*, vol. 25, págs. 28-30, junio de 2002.

————"The Eternity Service", *Proc. Pragocrypt Conf.*, CTU Publishing House, págs. 242-252, 1996.

ANDREWS, J., GHOSH, A. y MUHAMED, R. *Fundamentals of WiMAX: Understanding Broadband Wireless Networking*, Upper Saddle River, Pearson Educación, 2007.

ASTELY, D., DAHLMAN, E., FURUSKAR, A., JADING, Y., LINDSTROM, M. y PARKVALL, S. "LTE: The Evolution of Mobile Broadband", *IEEE Commun. Magazine*, vol. 47, págs. 44-51, abril de 2009.

BALLARDIE, T., FRANCIS, P y CROWCROFT, J. "Core Based Trees (CBT)", *Proc. SIGCOMM '93 Conf.*, ACM, págs. 85-95, 1993.

BARAN, P. "On Distributed Communications: I. Introduction to Distributed Communication Networks", *Memorandum RM-420-PR*, Rand Corporation, agosto de 1964.

BELLAMY, J. *Digital Telephony*, 3a. ed., Nueva York, John Wiley & Sons, 2000.

BELLMAN, R.E. *Dynamic Programming*, Princeton, NJ: Princeton University Press, 1957.

BELLOVIN, S. "The Security Flag in the IPv4 Header", RFC 3514, abril de 2003.

BELSNES, D. "Flow Control in the Packet Switching Networks", *Communications Networks*, Uxbridge, Inglaterra, en línea, págs. 349-361, 1975.

BENNET, C.H. y BRASSARD, G. "Quantum Cryptography: Public Key Distribution and Coin Tossing", *Int'l Conf. on Computer Systems and Signal Processing*, págs. 175-179, 1984.

BERESFORD, A. y STAJANO, F. "Location Privacy in Pervasive Computing", *IEEE Pervasive Computing*, vol. 2, págs. 46-55, enero de 2003.

BERGHEL, H.L. "Cyber Privacy in the New Millenium", *IEEE Computer*, vol. 34, págs. 132-134, enero de 2001.

BERNERS-LEE, T., CAILLIAU, A., LOUTONEN, A., NIELSEN, H.F. y SECRET, A. "The World Wide Web", *Commun. of the ACM,* vol. 37, págs. 76-82, agosto de 1994.

BERTSEKAS, D. y GALLAGER, R. *Data Networks*, 2a. ed., Englewood Cliffs, NJ: Prentice Hall, 1992.

BHATTI, S.N. y CROWCROFT, J. "QoS Sensitive Flows: Issues in IP Packet Handling", *IEEE Internet Computing,* vol. 4, págs. 48-57, julio-agosto de 2000.

BIHAM, E. y SHAMIR, A. "Differential Fault Analysis of Secret Key Cryptosystems", *Proc. 17th Ann. Int'l Cryptology Conf.,* Berlin, Springer-Verlag LNCS 1294, págs. 513-525, 1997.

BIRD, R., GOPAL, I., HERZBERG, A., JANSON, P.A., KUTTEN, S., MOLVA, R. y YUNG, M. "Systematic Design of a Family of Attack-Resistant Authentication Protocols", *IEEE J. on Selected Areas in Commun.*, vol. 11, págs. 679-693, junio de 1993.

BIRRELL, A.D. y NELSON, B.J. "Implementing Remote Procedure Calls", *ACM Trans. on Computer Systems*, vol. 2, págs. 39-59, febrero de 1984.

BIRYUKOV, A., SHAMIR, A. y WAGNER, D. "Real Time Cryptanalysis of A5/1 on a PC", *Proc. Seventh Int'l Workshop on Fast Software Encryption*, Berlin, Springer-Verlag LNCS 1978, págs. 1-8, 2000.

BLAZE, M. y BELLOVIN, S. "Tapping on My Network Door", *Commun. of the ACM*, vol. 43, pág. 136, octubre de 2000.

BOGGS, D., MOGUL, J. y KENT, C. "Measured Capacity of an Ethernet: Myths and Reality", *Proc. SIGCOMM '88 Conf.*, ACM, págs. 222-234, 1988.

BORISOV, N., GOLDBERG, I. y WAGNER, D. "Intercepting Mobile Communications: The Insecurity of 802.11", *Seventh Int'l Conf. on Mobile Computing and Networking*, ACM, págs. 180-188, 2001.

BRADEN, R. "Requirements for Internet Hosts-Communication Layers", RFC 1122, octubre de 1989.

BRADEN, R., BORMAN, D. y PARTRIDGE, C., "Computing the Internet Checksum", RFC 1071, septiembre de 1988.

BRANDENBURG, K. "MP3 and AAC Explained", *Proc. 17th Intl. Conf.: High-Quality Audio Coding,* Audio Engineering Society, págs. 99-110, agosto de 1999.

BRAY, T., PAOLI, J., SPERBERG-MCQUEEN, C., MALER, E., YERGEAU, F. y COWAN, J. "Extensible Markup Language (XML) 1.1 (Second Edition)", Recomendación del W3C, septiembre de 2006.

BRESLAU, L, CAO, P., FAN, L., PHILLIPS, G. y SHENKER, S. "Web Caching and Zipf-like Distributions: Evidence and Implications", *Proc. INFOCOM Conf.*, IEEE, págs. 126-134, 1999.

BURLEIGH, S., HOOKE, A., TORGERSON, L., FALL, K., CERF, V., DURST, B., SCOTT, K. y WEISS, H. "Delay-Tolerant Networking: An Approach to Interplanetary Internet", *IEEE Commun. Magazine*, vol. 41, págs. 128-136, junio de 2003.

BURNETT, S. y PAINE, S. *RSA Security's Official Guide to Cryptography*, Berkeley, CA: Osborne/McGraw-Hill, 2001.

BUSH, V. "As We May Think", *Atlantic Monthly*, vol. 176, págs. 101-108, julio de 1945.

CAPETANAKIS, J.I. "Tree Algorithms for Packet Broadcast Channels", *IEEE Trans. on Information Theory*, vol. IT-5, págs. 505-515, septiembre de 1979.

CASTAGNOLI, G., BRAUER, S. y HERRMANN, M. "Optimization of Cyclic Redundancy-Check Codes with 24 and 32 Parity Bits", *IEEE Trans. on Commun.*, vol. 41, págs. 883-892, junio de 1993.

CERF, V. y KAHN, R. "A Protocol for Packet Network Interconnection", *IEEE Trans. on Commun*, vol. COM-2, págs. 637-648, mayo de 1974.

CHANG, F., DEAN, J., GHEMAWAT, S., HSIEH, W., WALLACH, D., BURROWS, M., CHANDRA, T., FIKES, A. y GRUBER, R. "Bigtable: A Distributed Storage System for Structured Data", *Proc. OSDI 2006 Symp*., USENIX, págs. 15-29, 2006.

CHASE, J.S., GALLATIN, A.J. y YOCUM, K.G. "End System Optimizations for High-Speed TCP", *IEEE Commun. Magazine*, vol. 39, págs. 68-75, abril de 2001.

CHEN, S. y NAHRSTEDT, K. "An Overview of QoS Routing for Next-Generation Networks", *IEEE Network Magazine*, vol. 2, págs. 64-69, noviembre/diciembre de 1998.

CHIU, D. y JAIN, R. "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks", *Comput. Netw. ISDN Syst.*, vol. 17, págs. 1-4, junio de 1989.

CISCO. "Cisco Visual Networking Index: Forecast and Methodology, 2009-2014", Cisco Systems Inc., junio de 2010.

CLARK, D. D. "The Design Philosophy of the DARPA Internet Protocols", *Proc. SIGCOMM '88 Conf.*, ACM, págs. 106-114, 1988.

————"Window and Acknowledgement Strategy in TCP", RFC 813, julio de 1982.

CLARK, D.D., JACOBSON, V., ROMKEY, J. y SALWEN, H. "An Analysis of TCP Processing Overhead", *IEEE Commun. Magazine*, vol. 27, págs. 23-29, junio de 1989.

CLARK, D.D., SHENKER, S. y ZHANG, L. "Supporting Real-Time Applications in an Integrated Services Packet Network", *Proc. SIGCOMM '92 Conf.*, ACM, págs. 14-26, 1992.

CLARKE, A.C. "Extra-Terrestrial Relays", *Wireless World*, 1945.

CLARKE, I., MILLER, S. G., HONG, T. W., SANDBERG, O. y WILEY, B. "Protecting Free Expression Online with Freenet", *IEEE Internet Computing*, vol. 6, págs. 40-49, enero-febrero de 2002.

COHEN, B. “Incentives Build Robustness in BitTorrent”, *Proc. First Workshop on Economics of Peer-to-Peer Systems*, junio de 2003.

COMER, D. E. *The Internet Book*, 4a. ed., Englewood Cliffs, NJ: Prentice Hall, 2007.

———*Internetworking with TCP/IP*, vol. 1, 5a. ed., Englewood Cliffs, NJ: Prentice Hall, 2005.

CRAVER, S. A., WU, M., LIU, B., STUBBLEFIELD, A., SWARTZLANDER, B., WALLACH, D.W., DEAN, D. y FELTEN, E. W. “Reading Between the Lines: Lessons from the SDMI Challenge”, *Proc. 10th USENIX Security Symp.*, USENIX, 2001.

CROVELLA, M. y KRISHNAMURTHY, B. *Internet Measurement*, Nueva York, John Wiley & Sons, 2006.

DAEMEN, J. y RIJMEN, V. *The Design of Rijndael*, Berlin, Springer-Verlag, 2002.

DALAL, Y. y METCLFE, R. “Reverse Path Forwarding of Broadcast Packets”, *Commun. of the ACM*, vol. 21, págs. 1040-1048, diciembre de 1978.

DAVIE, B. y FARREL, A. *MPLS: Next Steps*, San Francisco, Morgan Kaufmann, 2008.

DAVIE, B. y REKHTER, Y. *MPLS Technology and Applications*, San Francisco, Morgan Kaufmann, 2000.

DAVIES, J. *Understanding IPv6*, 2a. ed., Redmond, WA: Microsoft Press, 2008.

DAY, J.D. “The (Un)Revised OSI Reference Model”, *Computer Commun. Rev.*, vol. 25, págs. 39-55, octubre de 1995.

DAY, J. D. y ZIMMERMANN, H. “The OSI Reference Model”, *Proc. Of the IEEE*, vol. 71, págs. 1334-1340, diciembre de 1983.

DECANDIA, G., HASTORIN, D., JAMPANI, M., KAKULAPATI, G., LAKSHMAN, A., PILCHIN, A., SIVASUBRAMANIAN, S., VOSSHALL, P. y VOGELS, W., “Dynamo: Amazon’s Highly Available Key-value Store”, *Proc. 19th Symp. on Operating Systems Prin.*, ACM, págs. 205-220, diciembre de 2007.

DEERING, S.E. “SIP: Simple Internet Protocol”, *IEEE Network Magazine*, vol. 7, págs. 16-28, mayo/junio de 1993.

DEERING, S. y CHERITON, D. “Multicast Routing in Datagram Networks and Extended LANs”, *ACM Trans. on Computer Systems*, vol. 8, págs. 85-110, mayo de 1990.

DEMERS, A., KESHAV, S. y SHENKER, S. “Analysis and Simulation of a Fair Queueing Algorithm”, *Internetwork: Research and Experience*, vol. 1, págs. 3-26, septiembre de 1990.

DENNING, D.E. y SACCO, G.M. “Timestamps in Key Distribution Protocols”, *Commun. of the ACM*, vol. 24, págs. 533-536, agosto de 1981.

DEVARAPALLI, V., WAKIKAWA, R., PETRESCU, A. y THUBERT, P. “Network Mobility (NEMO) Basic Support Protocol”, RFC 3963, enero de 2005.

DIFFIE, W. y HELLMAN, M. E. “Exhaustive Cryptanalysis of the NBS Data Encryption Standard”, *IEEE Computer*, vol. 10, págs. 74-84, junio de 1977.

———“New Directions in Cryptography”, *IEEE Trans. on Information Theory*, vol. IT-2, págs. 644-654, noviembre de 1976.

DIJKSTRA, E. W. “A Note on Two Problems in Connexion with Graphs”, *Numer. Math.*, vol. 1, págs. 269-271, octubre de 1959.

DILLEY, J., MAGGS, B., PARIKH, J., PROKOP, H., SITARAMAN, R. y WHEIL, B. "Globally Distributed Content Delivery", *IEEE Internet Computing*, vol. 6, págs. 50-58, 2002.

DINGLEDINE, R., MATHEWSON, N., SYVERSON, P. "Tor: The Second-Generation Onion Router", *Proc. 13th USENIX Security Symp.*, USENIX, pp. 303-320, agosto 2004.

DONAHOO, M. y CALVERT, K. *TCP/IP Sockets in C*, 2a. ed., San Francisco, Morgan Kaufmann, 2009.

DONAHOO, M. y CALVERT, K. *TCP/IP Sockets in Java*, 2a. ed., San Francisco, Morgan Kaufmann, 2008.

DONALDSON, G. y JONES, D. "Cable Television Broadband Network Architectures", *IEEE Commun. Magazine*, vol. 39, págs. 122-126, junio de 2001.

DORFMAN, R. "Detection of Defective Members of a Large Population", *Annals Math. Statistics*, vol. 14, págs. 436-440, 1943.

DUTCHER, B. *The NAT Handbook*, Nueva York, John Wiley & Sons, 2001.

DUTTA-ROY, A. "An Overview of Cable Modem Technology and Market Perspectives", *IEEE Commun. Magazine*, vol. 39, págs. 81-88, junio de 2001.

EDELMAN, B., OSTROVSKY, M. y SCHWARZ, M. "Internet Advertising and the Generalized Second-Price Auction: Selling Billions of Dollars Worth of Keywords", *American Economic Review*, vol. 97, págs. 242-259, marzo de 2007.

EL GAMAL, T. "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", *IEEE Trans. on Information Theory*, vol. IT-1, págs. 469-472, julio de 1985.

EPCGLOBAL. *EPC Radio-Frequency Identity Protocols Class- Generation- UHF RFID Protocol for Communication at 860-MHz to 960-MHz Version 1.2.0,* Bruselas, EPCglobal Inc., octubre de 2008.

FALL, K. "A Delay-Tolerant Network Architecture for Challenged Internets", *Proc. SIGCOMM 2003 Conf.*, ACM, págs. 27-34, agosto de 2003.

FALOUTSOS, M., FALOUTSOS, P. y FALOUTSOS, C. "On Power-Law Relationships of the Internet Topology", *Proc. SIGCOMM '99 Conf.*, ACM, págs. 251-262, 1999.

FARRELL, S. y CAHILL, V.: *Delay- and Disruption-Tolerant Networking*, Londres: Artech House, 2007.

FELLOWS, D. y JONES, D. "DOCSIS Cable Modem Technology", *IEEE Commun. Magazine*, vol. 39, págs. 202-209, marzo de 2001.

FENNER, B., HANDLEY, M., HOLBROOK, H. y KOUVELAS, I. "Protocol Independent Multicast-Sparse Mode (PIM-SM)", RFC 4601, agosto de 2006.

FERGUSON, N., SCHNEIER, B. y KOHNO, T., *Cryptography Engineering: Design Principles and Practical Applications*, Nueva York, John Wiley & Sons, 2010.

FLANAGAN, D. *JavaScript: The Definitive Guide*, 6a. ed., Sebastopol, CA: O'Reilly, 2010.

FLETCHER, J. "An Arithmetic Checksum for Serial Transmissions", *IEEE Trans. on Commun.*, vol. COM-0, págs. 247-252, enero de 1982.

FLOYD, S., HANDLEY, M., PADHYE, J. y WIDMER, J. "Equation-Based Congestion Control for Unicast Applications", *Proc. SIGCOMM 2000 Conf.*, ACM, págs. 43-56, agosto, 2000.

FLOYD, S. y JACOBSON, V. "Random Early Detection for Congestion Avoidance", *IEEE/ACM Trans. on Networking*, vol. 1, págs. 397-413, agosto de 1993.

FLUHRER, S., MANTIN, I. y SHAMIR, A. "Weakness in the Key Scheduling Algorithm of RC4", *Proc. Eighth Ann. Workshop on Selected Areas in Cryptography*, Berlin, Springer-Verlag LNCS 2259, págs. 1-24, 2001.

FORD, B. "Structured Streams: A New Transport Abstraction", *Proc. SIGCOMM 2007 Conf.*, ACM, págs. 361-372, 2007.

FORD, L.R., Jr. y FULKERSON, D. R. *Flows in Networks*, Princeton, NJ: Princeton University Press, 1962.

FORD, W. y BAUM, M. S. *Secure Electronic Commerce*, Upper Saddle River, NJ: Prentice Hall, 2000.

FORNEY, G. D. "The Viterbi Algorithm", *Proc. of the IEEE*, vol. 61, págs. 268-278, marzo de 1973.

FOULI, K. y MALER, M. "The Road to Carrier-Grade Ethernet", *IEEE Commun. Magazine*, vol. 47, págs. S30-S38, marzo de 2009.

FOX, A., GRIBBLE, S., BREWER, E. y AMIR, E. "Adapting to Network and Client Variability via On-Demand Dynamic Distillation", *SIGOPS Oper. Syst. Rev.*, vol. 30, págs. 160-170, diciembre de 1996.

FRANCIS, P. "A Near-Term Architecture for Deploying Pip", *IEEE Network Magazine*, vol. 7, págs. 30-37, mayo/junio de 1993.

FRASER, A. G. "Towards a Universal Data Transport System", *IEEE J. on Selected Areas in Commun.*, vol. 5, págs. 803-816, noviembre de 1983.

FRIDRICH, J. *Steganography in Digital Media: Principles, Algorithms, and Applications*, Cambridge, Cambridge University Press, 2009.

FULLER, V. y LI, T. "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", RFC 4632, agosto de 2006.

GALLAGHER, R.G. "A Minimum Delay Routing Algorithm Using Distributed Computation", *IEEE Trans. on Commun.*, vol. COM-5, págs. 73-85, enero de 1977.

GALLAGHER, R.G. "Low-Density Parity Check Codes", *IRE Trans. on Information Theory*, vol. 8, págs. 21-28, enero de 1962.

GARFINKEL, S. con SPAFFORD, G. *Web Security, Privacy and Commerce*, Sebastopol, CA: O'Reilly, 2002.

GAST, M. *802.11 Wireless Networks: The Definitive Guide*, 2a. ed., Sebastopol, CA: O'Reilly, 2005.

GERSHENFELD, N., KRIKORIAN, R. y COHEN, D. "The Internet of Things", *Scientific American*, vol. 291, págs. 76-81, octubre de 2004.

GILDER, G. "Metcalfe's Law and Legacy", *Forbes ASAP*, 13 de septiembre de 1993.

GOODE, B. "Voice over Internet Protocol", *Proc. of the IEEE*, vol. 90, págs. 1495-1517, septiembre de 2002.

GORALSKI, W. J. *SONET*, 2a. ed., Nueva York, McGraw-Hill, 2002.

GRAYSON, M., SHATZKAMER, K. y WAINNER, S. *IP Design for Mobile Networks*, Indianápolis, IN: Cisco Press, 2009.

GROBE, K. y ELBERS, J. "PON in Adolescence: From TDMA to WDM-PON", *IEEE Commun. Magazine*, vol. 46, págs. 26-34, enero de 2008.

GROSS, G., KAYCEE, M., LIN. A, MALIS, A. y STEPHENS, J. "The PPP Over AAL5", RFC 2364, julio de 1998.

HA, S., RHEE, I. y LISONG, X. "CUBIC: A New TCP-Friendly High-Speed TCP Variant", *SIGOPS Oper. Syst. Rev.,* vol.42, págs. 64-74, junio de 2008.

HAFNER, K. y LYON, M. *Where Wizards Stay Up Late*, Nueva York, Simon & Schuster, 1998.

HALPERIN, D., HEYDT-BENJAMIN, T., RANSFORD, B., CLARK, S., DEFEND, B., MORGAN, W., FU, K., KOHNO, T. y MAISEL, W. "Pacemakers and Implantable Cardiac Defibrillators: Software Radio Attacks and Zero-Power Defenses", *IEEE Symp. on Security and Privacy*, págs. 129-142, mayo de 2008.

HALPERIN, D., HU, W., SHETH, A. y WETHERALL, D. "802.11 with Multiple Antennas for Dummies", *Computer Commun. Rev.*, vol. 40, págs. 19-25, enero de 2010.

HAMMING, R.W. "Error Detecting and Error Correcting Codes", *Bell System Tech. J.*, vol. 29, págs. 147-160, abril de 1950.

HARTE, L., KELLOGG, S., DREHER, R. y SCHAFFNIT, T. *The Comprehensive Guide to Wireless Technology*, Fuquay-Varina, NC: APDG Publishing, 2000.

HAWLEY, G.T. "Historical Perspectives on the U.S. Telephone Loop", *IEEE Commun. Magazine*, vol. 29, págs. 24-28, marzo de 1991.

HECHT, J. *Understanding Fiber Optics*, Upper Saddle River, NJ: Prentice Hall, 2005.

HELD, G. *A Practical Guide to Content Delivery Networks*, 2a. ed., Boca Ratón, FL: CRC Press, 2010.

HEUSSE, M., ROUSSEAU, F., BERGER-SABBATEL, G., DUDA, A. "Performance Anomaly of 802.11b", *Proc. INFOCOM Conf.*, IEEE, págs. 836-843, 2003.

HIERTZ, G., DENTENEER, D., STIBOR, L., ZANG, Y., COSTA, X. y WALKE, B. "The IEEE 802.11 Universe", *IEEE Commun. Magazine*, vol. 48, págs. 62-70, enero de 2010.

HOE, J. "Improving the Start-up Behavior of a Congestion Control Scheme for TCP", *Proc. SIGCOMM '96 Conf.*, ACM, págs. 270-280, 1996.

HU, Y. y LI, V.O.K. "Satellite-Based Internet: A Tutorial", *IEEE Commun. Magazine*, vol. 30, págs. 154-162, marzo de 2001.

HUITEMA, C. *Routing in the Internet*, 2a. ed., Englewood Cliffs, NJ: Prentice Hall, 1999.

HULL, B., BYCHKOVSKY, V., CHEN, K., GORACZKO, M., MIU, A., SHIH, E., ZHANG, Y., BALAKRISHNAN, H. y MADEN, S. "CarTel: A Distributed Mobile Sensor Computing System", *Proc. Sensys 2006 Conf.*, ACM, págs. 125-138, noviembre de 2006.

HUNTER, D., RAFTER, J., FAWCETT, J., VAN DER LIST, E., AYERS, D., DUCKETT, J., WATT, A. y MCKINNON, L. *Beginning XML*, 4a. ed., Nueva Jersey, Wrox, 2007.

IRMER, T. "Shaping Future Telecommunications: The Challenge of Global Standardization", *IEEE Commun. Magazine*, vol. 32, págs. 20-28, enero de 1994.

ITU (INTERNATIONAL TELECOMMUNICATION UNION): *ITU Internet Reports 2005: The Internet of Things*, Ginebra, ITU, noviembre de 2005.

————*Measuring the Information Society: The ICT Development Index,* Ginebra, ITU, marzo de 2009.

JACOBSON, V. "Compressing TCP/IP Headers for Low-Speed Serial Links", RFC 1144, febrero de 1990.

————"Congestion Avoidance and Control", *Proc. SIGCOMM '88 Conf.*, ACM, págs. 314-329, 1988.

JAIN, R. y ROUTHIER, S. “Packet Trains-Measurements and a New Model for Computer Network Trafic”, *IEEE J. on Selected Areas in Commun.*, vol. 6, págs. 986-995, septiembre de 1986.

JAKOBSSON, M. y WETZEL, S. “Security Weaknesses in Bluetooth”, *Topics in Cryptology: CT-RSA 2001*, Berlin, Springer-Verlag LNCS 2020, págs. 176-191, 2001.

JOEL, A. “Telecommunications and the IEEE Communications Society”, *IEEE Commun. Magazine*, 50th Anniversary Issue, págs. 6-14 y 162-167, mayo de 2002.

JOHNSON, D., PERKINS, C. y ARKKO, J. “Mobility Support in Ipv6”, RFC 3775, junio de 2004.

JOHNSON, D. B., MALTZ, D. y BROCH, J. “DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks”, *Ad Hoc Networking*, Boston, Addison-Wesley, págs. 139-172, 2001.

JUANG, P., OKI, H., WANG Y., MARTONOSI, M., PEH, L, y RUBENSTEIN, D. “Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet”, *SIGOPS Oper. Syst. Rev.*, vol. 36, págs. 96-107, octubre de 2002.

KAHN, D. *The Codebreakers*, 2a. ed., Nueva York, Macmillan, 1995.

KAMOUN, F. y KLEINROCK, L. “Stochastic Performance Evaluation of Hierarchical Routing for Large Networks”, *Computer Networks*, vol. 3, págs. 337-353, noviembre de 1979.

KARN, P. “MACA-A New Channel Access Protocol for Packet Radio”, *ARRL/CRRL Amateur Radio Ninth Computer Networking Conf.*, págs. 134-140, 1990.

KARN, P. y PARTRIDGE, C. “Improving Round-Trip Estimates in Reliable Transport Protocols”, *Proc. SIGCOMM ’87 Conf.*, ACM, págs. 2-7, 1987.

KARP, B. y KUNG, H.T. “GPSR: Greedy Perimeter Stateless Routing for Wireless Networks”, *Proc. MOBICOM 2000 Conf.*, ACM, págs. 243-254, 2000.

KASIM, A. *Delivering Carrier Ethernet*, Nueva York, McGraw-Hill, 2007.

KATABI, D., HANDLEY, M. y ROHRS, C. “Internet Congestion Control for Future High Bandwidth-Delay Product Environments”, *Proc. SIGCOMM 2002 Conf.*, ACM, págs. 89-102, 2002.

KATZ, D. y FORD, P. S. “TUBA: Replacing IP with CLNP”, *IEEE Network Magazine*, vol. 7, págs. 38-47, mayo/junio de 1993.

KAUFMAN, C., PERLMAN, R. y SPECINER, M., *Network Security*, 2a. ed., Englewood Cliffs, NJ: Prentice Hall, 2002.

KENT, C. y MOGUL, J. “Fragmentation Considered Harmful”, *Proc. SIGCOMM ’87 Conf.*, ACM, págs. 390-401, 1987.

KERCKHOFF, A. “La Cryptographie Militaire”, *J. des Sciences Militaires*, vol. 9, págs. 5-38, enero de 1883 y págs. 161-191, febrero de 1883.

KHANNA, A. y ZINKY, J. “The Revised ARPANET Routing Metric”, *Proc. SIGCOMM ’89 Conf.*, ACM, págs. 45-56, 1989.

KIPNIS, J. “Beating the System: Abuses of the Standards Adoption Process”, *IEEE Commun. Magazine*, vol. 38, págs. 102-105, julio de 2000.

KLEINROCK, L. “Power and Other Deterministic Rules of Thumb for Probabilistic Problems in Computer Communications”, *Proc. Intl. Conf. on Commun.*, págs. 43.1.1-43.1.10, junio de 1979.

KLEINROCK, L. y TOBAGI, F. “Random Access Techniques for Data Transmission over Packet-Switched Radio Channels”, *Proc. Nat. Computer Conf.*, págs. 187-201, 1975.

KOHLER, E., HANDLEY, H. y FLOYD, S. "Designing DCCP: Congestion Control without Reliability", *Proc. SIGCOMM 2006 Conf.*, ACM, págs. 27-38, 2006.

KOODLI, R. y PERKINS, C. E. *Mobile Inter-networking with IPv6*, Nueva York, John Wiley & Sons, 2007.

KOOPMAN, P. "32-Bit Cyclic Redundancy Codes for Internet Applications", *Proc. Intl. Conf. on Dependable Systems and Networks*, IEEE, págs. 459-472, 2002.

KRISHNAMURTHY, B. y REXFORD, J. *Web Protocols and Practice*, Boston, Addison-Wesley, 2001.

KUMAR, S., PAAR, C., PELZL, J., PFEIFFER, G. y SCHIMMLER, M. "Breaking Ciphers with COPACOBANA: A Cost-Optimized Parallel Code Breaker", *Proc. 8th Cryptographic Hardware and Embedded Systems Wksp.*, IACR, págs. 101-118, octubre de 2006.

LABOVITZ, C., AHUJA, A., BOSE, A. y JAHANIAN, F. "Delayed Internet Routing Convergence", *IEEE/ACM Trans. on Networking*, vol. 9, págs. 293-306, junio de 2001.

LAM, C.K.M y TAN, B.C.Y. "The Internet is Changing the Music Industry", *Commun. of the ACM*, vol. 44, págs. 62-66, agosto de 2001.

LAOUTARIS, N., SMARAGDAKIS, G., RODRIGUEZ, P. y SUNDARAM, R. "Delay Tolerant Bulk Data Transfers on the Internet", *Proc. SIGMETRICS 2009 Conf.*, ACM, págs. 229-238, junio 2009.

LARMO, A., LINDSTROM, M., MEYER, M., PELLETIER, G., TORSNER, J. y WIEMANN, H. "The LTE Link-Layer Design", *IEEE Commun. Magazine*, vol. 47, págs. 52-59, abril de 2009.

LEE, J.S. y MILLER, L.E. *CDMA Systems Engineering Handbook*, Londres, Artech House, 1998.

LELAND, W., TAQQU, M., WILLINGER, W. y WILSON, D. "On The Self-Similar Nature of Ethernet Traffic", *IEEE/ACM Trans. on Networking*, vol. 2, págs. 1-15, febrero de 1994.

LEMON, J. "Resisting SYN Flood DOS Attacks with a SYN Cache", *Proc. BSDCon Conf.*, USENIX, págs. 88-98, 2002.

LEVY, S. "Crypto Rebels", *Wired*, págs. 54-61, mayo/junio 1993.

LEWIS, M. *Comparing, Designing and Deploying VPNs*, Indianápolis, IN: Cisco Press, 2006.

LI, M., AGRAWAL, D., GANESAN, D. y VENKATARAMANI, A. "Block-Switched Networks: A New Paradigm for Wireless Transport", *Proc. NSDI 2009 Conf.*, USENIX, págs. 423-436, 2009.

LIN, S. y COSTELLO, D. *Error Control Coding*, 2a. ed., Upper Saddle River, NJ: Pearson Education, 2004.

LUBACZ, J., MAZURCZYK, W. y SZCZYPIORSKI, K. "Vice over IP", *IEEE Spectrum*, págs. 42-47, febrero de 2010.

MACEDONIA, M.R. "Distributed File Sharing", *IEEE Computer*, vol. 33, págs. 99-101, 2000.

MADHAVAN, J., KO, D., LOT, L., GANGPATHY, V., RASMUSSEN, A. y HALEVY, A. "Google's Deep Web Crawl", *Proc. VLDB 2008 Conf.*, VLDB Endowment, págs. 1241-1252, 2008.

MAHAJAN, R., RODRIG, M., WETHERALL, D. y ZAHORJAN, J. "Analyzing the MAC-Level Behavior of Wireless Networks in the Wild", *Proc. SIGCOMM 2006 Conf.*, ACM, págs. 75-86, 2006.

MALIS, A. y SIMPSON, W. "PPP over SONET/SDH", RFC 2615, junio de 1999.

MASSEY, J.L. "Shift-Register Synthesis and BCH Decoding", *IEEE Trans. on Information Theory*, vol. IT-5, págs. 122-127, enero de 1969.

MATSUI, M. "Linear Cryptanalysis Method for DES Cipher", *Advances in Cryptology-Eurocrypt 1993 Proceedings*, Berlin, Springer-Verlag LNCS 765, págs. 386-397, 1994.

MAUFER, T.A. *IP Fundamentals*, Upper Saddle River, NJ: Prentice Hall, 1999.

MAYMOUNKOV, P. y MAZIERES, D. "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric", *Proc. First Intl. Wksp. on Peer-to-Peer Systems*, Berlin, Springer-Verlag LNCS 2429, págs. 53-65, 2002.

MAZIERES, D. y KAASHOEK, M.F. "The Design, Implementation and Operation of an Email Pseudonym Server", *Proc. Fifth Conf. on Computer and Commun. Security*, ACM, págs. 27-36, 1998.

MCAFEE LABS. *McAfee Threat Reports: First Quarter 2010*, McAfee, Inc., 2010.

MENEZES, A. J. y VANSTONE, S. A. "Elliptic Curve Cryptosystems and Their Implementation", *Journal of Cryptology*, vol. 6, págs. 209-224, 1993.

MERKLE, R. C. y HELLMAN, M. "Hiding and Signatures in Trapdoor Knapsacks", *IEEE Trans. on Information Theory*, vol. IT-4, págs. 525-530, septiembre de 1978.

METCALFE, R. M. "Computer/Network Interface Design: Lessons from Arpanet and Ethernet", *IEEE J. on Selected Areas in Commun.*, vol. 11, págs. 173-179, febrero de 1993.

METCALFE, R. M. y BOGGS, D. R. "Ethernet: Distributed Packet Switching for Local Computer Networks", *Commun. of the ACM*, vol. 19, págs. 395-404, julio de 1976.

METZ, C. "Interconnecting ISP Networks", *IEEE Internet Computing*, vol. 5, págs. 74-80, marzo-abril de 2001.

MISHRA, P. P., KANAKIA, H. y TRIPATHI, S. "On Hop by Hop Rate-Based Congestion Control", *IEEE/ACM Transf. on Networking*, vol. 4, págs. 224-239, abril de 1996.

MOGUL, J. C. "IP Network Performance", en *Internet System Handbook*, D.C., Lynch y M.Y. Rose (editores), Boston, Addison-Wesley, págs. 575-575, 1993.

MOGUL, J. y DEERING, S. "Path MTU Discovery", RFC 1191, noviembre de 1990.

MOGUL, J. y MINSHALL, G. "Rethinking the Nagle Algorithm", *Comput. Commun. Rev.*, vol. 31, págs. 6-20, enero de 2001.

MOY, J. "Multicast Routing Extensions for OSPF", *Commun. of the ACM*, vol. 37, págs. 61-66, agosto de 1994.

MULLINS, J. "Making Unbreakable Code", *IEEE Spectrum*, págs. 40-45, mayo de 2002.

NAGLE, J. "On Packet Switches with Infinite Storage", *IEEE Trans. on Commun.*, vol. COM-5, págs. 435-438, abril de 1987.

————"Congestion Control in TCP/IP Internetworks", *Computer Commun. Rev.*, vol. 14, págs. 11-17, octubre de 1984.

NAUGHTON, J. *A Brief History of the Future*, Woodstock, NY: Overlook Press, 2000.

NEEDHAM, R. M. y SCHROEDER, M. D. "Using Encryption for Authentication in Large Networks of Computers", *Commun. of the ACM*, vol. 21, págs. 993-999, diciembre de 1978.

NEEDHAM, R. M. y SCHROEDER, M. D. "Authentication Revisited", *Operating Systems Rev.*, vol. 21, pág. 7, enero de 1987.

NELAKUDITI, S. y ZHANG, Z.-L. "A Localized Adaptive Proportioning Approach to QoS Routing", *IEEE Commun. Magazine*, vol. 40, págs. 66-71, junio de 2002.

NEUMAN, C. y TS'O, T. "Kerberos: An Authentication Service for Computer Networks", *IEEE Commun. Mag.*, vol. 32, págs. 33-38, septiembre de 1994.

NICHOLS, R. K. y LEKKAS, P. C. *Wireless Security*, Nueva York, McGraw-Hill, 2002.

NIST. "Secure Hash Algorithm", Estándar de procesamiento de información del gobierno federal de Estados Unidos 180, 1993.

NONNEMACHER, J., BIERSACK, E. y TOWSLEY, D. "Parity-Based Loss Recovery for Reliable Multicast Transmission", *Proc. SIGCOMM '97 Conf.*, ACM, págs. 289-300, 1997.

NUCCI, A. y PAPAGIANNAKI, D. *Design, Measurement and Management of Large-Scale IP Networks*, Cambridge, Cambridge University Press, 2008.

NUGENT, R., MUNAKANA, R., CHIN, A., COELHO, R. y PUIG-SUARI, J. "The CubeSat: The PicoSatellite Standard for Research and Education", *Proc. SPACE 2008 Conf.*, AIAA, 2008.

ORAN, D. "OSI IS-IS Intra-domain Routing Protocol", RFC 1142, febrero de 1990.

OTWAY, D. y REES, O., "Efficient and Timely Mutual Authentication", *Operating Systems Rev.*, págs. 8-10, enero de 1987.

PADHYE, J., FIROIU, V., TOWSLEY, D. y KUROSE, J. "Modeling TCP Throughput: A Simple Model and Its Empirical Validation", *Proc. SIGCOM '98 Conf.*, ACM, págs. 303-314, 1998.

PALAIS, J. C. *Fiber Optic Commun.*, 5a. ed., Englewood Cliffs, NJ: Prentice Hall, 2004.

PARAMESWARAN, M., SUSARLA, A. y WHINSTON, A. B. "P2P Networking: An Information-Sharing Alternative", *IEEE Computer*, vol. 34, págs. 31-38, julio de 2001.

PAREKH, A. y GALLAGHER, R. "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Multiple-Node Case", *IEEE/ACM Trans. on Networking*, vol. 2, págs. 137-150, abril de 1994.

PAREKH, A. y GALLAGHER, R. "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case", *IEEE/ACM Trans. on Networking*, vol. 1, págs. 344-357, junio de 1993.

PARTRIDGE, C., HUGHES, J. y STONE, J. "Performance of Checksums and CRCs over Real Data", *Proc. SIGCOMM '95 Conf.*, ACM, págs. 68-76, 1995.

PARTRIDGE, C., MENDEZ, T. y MILLIKEN, W. "Host Anycasting Service", RFC 1546, noviembre de 1993.

PAXSON, V. y FLOYD, S. "Wide-Area Traffic: The Failure of Poisson Modeling", *IEEE/ACM Trans. on Networking*, vol. 3, págs. 226-244, junio de 1995.

PERKINS, C. "IP Mobility Support for IPv4", RFC 3344, agosto de 2002.

PERKINS, C. E. *RTP: Audio and Video for the Internet*, Boston, Addison-Wesley, 2003.

PERKINS, C. E. (ed.). *Ad Hoc Networking*, Boston, Addison-Wesley, 2001.

PERKINS, C. E. *Mobile IP Design Principles and Practices*, Upper Saddle River, NJ: Prentice Hall, 1998.

PERKINS, C.E. y ROYER, E. “The Ad Hoc On-Demand Distance-Vector Protocol”, en *Ad Hoc Networking*, editado por C. Perkins, Boston, Addison-Wesley, 2001.

PERLMAN, R. *Interconnections*, 2a. ed., Boston, Addison-Wesley, 2000.

PERLMAN, R. *Network Layer Protocols with Byzantine Robustness*, tesis Ph.D., M.I.T., 1988.

PERLMAN, R. “An Algorithm for the Distributed Computation of a Spanning Tree in an Extended LAN”, *Proc. SIGCOMM ’85 Conf.*, ACM, págs. 44-53, 1985.

PERLMAN, R. y KAUFMAN, C. “Key Exchange in IPsec”, *IEEE Internet Computing*, vol. 4, págs. 50-56, noviembre-diciembre de 2000.

PETERSON, W. W. y BROWN, D.T. “Cyclic Codes for Error Detection”, *Proc. IRE*, vol. 49, págs. 228-235, enero de 1961.

PIATEK, M., KOHNO, T. y KRISHNAMURTHY, A. “Challenges and Directions for Monitoring P2P File Sharing Networks —or Why My Printer Received a DMCA Takedown Notice”, *3rd Workshop on Hot Topics in Security*, USENIX, julio de 2008.

PIATEK, M., ISDAL, T., ANDERSON, T., KRISHNAMURTHY, A. y VENKATARAMANI, V. “Do Incentives Build Robustness in BitTorrent?”, *Proc. NSDI 2007 Conf.*, USENIX, págs. 1-14, 2007.

PISCITELLO, D. M. y CHAPIN, A. L. *Open Systems Networking: TCP/IP and OSI*, Boston, Addison-Wesley, 1993.

PIVA, A., BAROLINI, F. y BARNI, M. “Managing Copyrights in Open Networks”, *IEEE Internet Computing*, vol. 6, págs. 18-26, mayo de 2002.

POSTEL, J. “Internet Control Message Protocols”, RFC 792, septiembre de 1981.

RABIN, J. y McCATHIENEVILE, C. “Mobile Web Best Practices 1.0”, Recomendación del W3C, julio de 2008.

RAMAKRISHNAM, K. K, FLOYD, S. y BLACK, D. “The Addition of Explicit Congestion Notification (ECN) to IP”, RFC 3168, septiembre de 2001.

RAMAKRISHNAN, K. K. y JAIN, R. “A Binary Feedback Scheme for Congestion Avoidance in Computer Networks with a Connectionless Network Layer”, *Proc. SIGCOMM ’88 Conf.*, ACM, págs. 303-313, 1988.

RAMASWAMI, R., KUMAR, S. y SASAKI, G. *Optical Networks: A Practical Perspective*, 3a. ed., San Francisco, Morgan Kaufmann, 2009.

RATNASAMY, S., FRANCIS, P., HANDLEY, M., KARP, R. y SHENKER, S. “A Scalable Content-Addressable Network”, *Proc. SIGCOMM 2001 Conf.*, ACM, págs. 161-172, 2001.

RIEBACK, M., CRISPO, B. y TANENBAUM, A. “Is Your Cat Infected with a Computer Virus?”, *Proc. IEEE Percom*, págs. 169-179, marzo de 2006.

RIVEST, R. L. “The MD5 Message-Digest Algorithm”, RFC 1320, abril de 1992.

RIVEST, R. L., SHAMIR, A. y ADLEMAN, L. “On a Method for Obtaining Digital Signatures and Public Key Cryptosystems”, *Commun. of the ACM*, vol. 21, págs. 120-126, febrero de 1978.

ROBERTS, L. G. “Extensions of Packet Communication Technology to a Hand Held Personal Terminal”, *Proc. Spring Joint Computer Conf.*, AFIPS, págs. 295-298, 1972.

———— “Multiple Computer Networks and Intercomputer Communication”, *Proc. First. Symp. on Operating Systems Prin.*, ACM, págs. 3.1-3.6, 1967.

ROSE, M.T. *The Simple Book*, Englewood Cliffs, NJ: Prentice Hall, 1994.

————*The Internet Message*, Englewood Cliffs, NJ: Prentice Hall, 1993.

ROWSTRON, A. y DRUSCHEL, P.: "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Storage Utility", *Proc. 18th Int'l Conf. on Distributed Systems Platforms*, Londres, Springer-Verlag LNCS 2218, págs. 329-350, 2001.

RUIZ-SANCHEZ, M. A., BIERSACK, E. W. y DABBOUS, W. "Survey and Taxonomy of IP Address Lookup Algorithms", *IEEE Network Magazine*, vol. 15, págs. 8-23, marzo-abril de 2001.

SALTZER, J. H., REED, D. P. y CLARK, D. D. "End-to-End Arguments in System Design", *ACM Trans. on Computer Systems*, vol. 2, págs. 277-288, noviembre de 1984.

SAMPLE, A., YEAGER, D., POWLEDGE, P., MAMISHEV, A. y SMITH, J. "Design of an RFID-Based Battery-Free Programmable Sensing Platform", *IEEE Trans. on Instrumentation and Measurement*, vol. 57, págs. 2608-2615, noviembre de 2008.

SAROIU, S., GUMMADI, K. y GRIBBLE, S. "Measuring and Analyzing the Characteristics of Napster & Gnutella Hosts", *Multim. Syst.*, vol. 9, págs. 170-184, agosto de 2003.

SCHALLER, R. "Moore's Law: Past, Present and Future", *IEEE Spectrum*, vol. 34, págs. 52-59, junio de 1997.

SCHNEIER, B. *Secrets and Lies*. Nueva York, John Wiley & Sons, 2004.

————*E-Mail Security*, Nueva York, John Wiley & Sons, 1995.

SCHNORR, C. P. "Efficient Signature Generation for Smart Cards", *Journal of Cryptology*, vol. 4, págs. 161-174, 1991.

SCHOLTZ, R. A. "The Origins of Spread-Spectrum Communications", *IEEE Trans. on Commun.*, vol. COM-0, págs. 822-854, mayo de 1982.

SCHWARTZ, M. y ABRAMSON, N. "The AlohaNet: Surfing for Wireless Data", *IEEE Commun. Magazine*, vol. 47, págs. 21-25, diciembre de 2009.

SEIFERT, R. y EDWARDS, J. *The All-New Switch Book*, NY: John Wiley, 2008.

SENN, J. A. "The Emergence of M-Commerce", *IEEE Computer*, vol. 33, págs. 148-150, diciembre de 2000.

SERJANTOV, A. "Anonymizing Censorship Resistant Systems", *Proc. First Int'l Workshop on Peer-to-Peer Systems*, Londres, Springer-Verlag LNCS 2429, págs. 111-120, 2002.

SCHACHAM, N. y MCKENNY, P. "Packet Recovery in High-Speed Networks Using Coding and Buffer Management", *Proc. INFOCOM Conf.*, IEEE, págs. 124-131, junio de 1990.

SHAIKH, A., REXFORD, J. y SHIN, K. "Load-Sensitive Routing of Long-Lived IP Flows", *Proc. SIGCOMM '99 Conf.*, ACM, págs. 215-226, septiembre 1999.

SHALUNOV, S. y CARLSON, R. "Detecting Duplex Mismatch on Ethernet", *Passive and Active Network Measurement*, Berlin, Springer-Verlag LNCS 3431, págs. 3135-3148, 2005.

SHANNON, C. "A Mathematical Theory of Communication", *Bell System Tech. J.*, vol. 27, págs. 379-423, julio de 1948; y págs. 623-656, octubre de 1948.

SHEPARD, S. *SONET/SDH Demystified*, Nueva York, McGraw-Hill, 2001.

SHREEDHAR, M. y VARGHESE, G. "Efficient Fair Queueing Using Deficit Round Robin", *Proc. SIGCOMM '95 Conf.*, ACM, págs. 231-243, 1995.

SIMPSON, W. *Video Over IP*, 2a. ed., Burlington, MA: Focal Press, 2008.

————“ PPP in HDLC-like Framing” RFC 1662, julio de 1994b

————“The Point-to-Point Protocol (PPP)”, RFC 1661, julio de 1994a.

SIU, K y JAIN, R. “A Brief Overview of ATM: Protocol Layers, LAN Emulation, and Traffic”, *ACM Computer Communications Review*, vol. 25, págs. 6-20, abril de 1995.

SKOUDIS, E. y LISTON, T. *Counter Hack Reloaded*, 2a. ed., Upper Saddle River, NJ: Prentice Hall, 2006.

SMITH, D. K. y ALEXANDER, R. C. *Fumbling the Future*, Nueva York, William Morrow, 1988.

SNOEREN, A. C. y BALAKRISHNAN, H. “An End-to-End Approach to Host Mobility”, *Int'l Conf. on Mobile Computing and Networking*, ACM, págs. 155-166, 2000.

SOBEL, D. L. “Will Carnivore Devour Online Privacy”, *IEEE Computer*, vol. 34, págs. 87-88, mayo de 2001.

SOTIROV, A., STEVENS, M., APPLEBAUM, J., LENSTRA, A., MOLNAR, D., OSVIK, D. y DE WEGER, B. “MD5 Considered Harmful Today”, *Proc. 25th Chaos Communication Congress*, Verlag Art d' Ameublement, 2008.

SOUTHEY, R. *The Doctors*, Londres, Longman, Brown, Green y Longmans, 1848.

SPURGEON, C. E. *Ethernet: The Definitive Guide*, Sebastopol, CA: O'Reilly, 2000.

STALLINGS, W. *Data and Computer Communications*, 9a. ed., Upper Saddle River, NJ: Pearson Education, 2010.

STARR, T., SORBARA, M., COIFFI, J. y SILVERMAN, P. “DSL Advances”, Upper Saddle River, NJ: Prentice Hall, 2003.

STEVENS, W. R. *TCP/IP Illustrated: The Protocols*, Boston, Addison Wesley, 1994.

STINSON, D. R. *Cryptography Theory and Practice*, 2a. ed., Boca Ratón, FL: CRC Press, 2002.

STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, M. F. y BALAKRISHNAN, H. “Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications”, *Proc. SIGCOMM 2001 Conf.*, ACM, págs. 149-160, 2001.

STUBBLEFIELD, A., IOANNIDIS, J. y RUBIN, A. D. “Using the Fluhrer, Mantin, and Shamir Attack to Break WEP”, *Proc. Network and Distributed Systems Security Symp.*, ISOC, págs. 1-11, 2002.

STUTTARD, D. y PINTO, M. *The Web Application Hacker's Handbook*, Nueva York, John Wiley & Sons, 2007.

SU, S. *The UMTS Air Interface in RF Engineering*, Nueva York, McGraw-Hill, 2007.

SULLIVAN, G. y WIEGAND, T. “Tree Algorithms for Packet Broadcast Channels”, *Proc. of the IEEE*, vol. 93, págs. 18-31, enero de 2005.

SUNSHINE, C. A. y DALAL, Y. K. “Connection Management in Transport Protocols”, *Computer Networks*, vol. 2, págs. 454-473, 1978.

TAN, K., SONG, J., ZHANG, Q. y SRIDHARN, M. “A Compound TCP Approach for High-Speed and Long Distance Networks”, *Proc. INFOCOM Conf.*, IEEE, págs. 1-12, 2006.

TANENBAUM, A. S. *Modern Operating Systems*, 3a. ed., Upper Saddle River, NJ: Prentice Hall, 2007.

TANENBAUM, A. S. y VAN STEEN, M. *Distributed Systems: Principles and Paradigms*, Upper Saddle River, NJ: Prentice Hall, 2007.

TOMLINSON, R. S. "Selecting Sequence Numbers", *Proc. SIGCOMM/SIGOPS Interprocess Commun. Workshop*, ACM, págs. 11-23, 1975.

TUCHMAN, W. "Hellman Presents No Shortcut Solutions to DES", *IEEE Spectrum*, vol. 16, págs. 40-41, julio de 1979.

TURNER, J. S. "New Directions in Communications (or Which Way to the Information Age)", *IEEE Commun. Magazine*, vol. 24, págs. 8-15, octubre de 1986.

UNGERBOECK, G. "Trellis-Coded Modulation with Redundant Signal Sets Part I: Introduction", *IEEE Commun. Magazine*, vol. 25, págs. 5-11, febrero de 1987.

VALADE, J. *PHP & MySQL for Dummies*, 5a. ed., Nueva York, John Wiley & Sons, 2009.

VARGHESE, G. *Network Algorithmics*, San Francisco, Morgan Kaufmann, 2004.

VARGHESE, G. y LAUCK, T. "Hashed and Hierarchical Timing Wheels: Data Structures for the Efficient Implementation of a Timer Facility", *Proc. 11th Symp. on Operating Systems Prin.*, ACM, págs. 25-38, 1987.

VERIZON BUSINESS. *2009 Data Breach Investigation Report*, Verizon, 2009.

VITERBI, A. *CDMA: Principles of Spread Spectrum Communication*, Englewood Cliffs, NJ: Prentice Hall, 1995.

VON AHN, L., BLUM, B. y LANGFORD, J. "Telling Humans and Computers Apart Automatically", *Commun. of the ACM*, vol. 47, págs. 56-60, febrero de 2004.

WAITZMAN, D., PARTRIDGE, C. y DEERING, S. "Distance Vector Multicast Routing Protocol", RFC 1075, noviembre de 1988.

WALDMAN, M., RUBIN, A. D. y CRANOR, L. F. "Publius: A Robust, Tamper-Evident, Censorship-Resistant Web Publishing System", *Proc. Ninth USENIX Security Symp.*, USENIX, págs. 59-72, 2000.

WANG, Z. y CROWCROFT, J.: "SEAL Detects Cell Misordering", *IEEE Network Magazine*, vol. 6, págs. 8-9, julio de 1992.

WANT, R. *RFID Explained*, San Rafael, CA: Morgan Claypool, 2006.

WARNEKE, B., LAST, M., LIEBOWITZ, B y PISTER, K. S. J. "Smart Dust: Communicating with a Cubic Millimeter Computer", *IEEE Computer*, vol. 34, págs. 44-51, enero de 2001.

WAYNER, P. *Disappearing Cryptography: Information Hiding, Steganography, and Watermarking*, 3a. ed., San Francisco, Morgan Kaufmann, 2008.

WEI, D., CHENG, J., LOW, S. y HEDGE, S. "FAST TCP: Motivation, Architecture, Algorithms, Performance", *IEEE/ACM Trans. on Networking*, vol. 14, págs. 1246-1259, diciembre 2006.

WEISER, M. "The Computer for the Twenty-First Century", *Scientific American*, vol. 262, págs. 94-104, septiembre de 1991.

WELBOURNE, E., BATTLE, L., COLE, G., GOULD, K., RECTOR, K., RAYMER, S., BALAZINSKA, M. y BORRIELLO, G. "Building the Internet of Things Using RFID", *IEEE Internet Computing*, vol. 13, págs. 48-55, mayo de 2009.

WITTENBURG, N. *Understanding Voice Over IP Technology*, Clifton Park, NY: Delmar Cengage Learning, 2009.

WOLMAN, A., VOELKER, G., SHARMA, N., CARDWELL, N., KARLIN, A. y LEVY, H. "On the Scale and Performance of Cooperative Web Proxy Caching", *Proc. 17th Symp. on Operating Systems Prin.*, ACM, págs. 16-31, 1999.

WOOD, L., IVANCIC, W., EDDY, W., STEWART, D., NORTHAM, J., JACKSON, C. y DA SILVA CURIEL, A. "Use of the Delay-Tolerant Networking Bundle Protocol from Space", *Proc. 59th Int'l Astronautical Congress*, Int'l Astronautical Federation, págs. 3123-3133, 2008.

WU, T. "Network Neutrality, Broadband Discrimination", *Journal on Telecom. and High-Tech. Law*, vol. 2, págs. 141-179, 2003.

WYLIE, J., BIGRIGG, M. W., STRUNK, J. D., GANGER, G. R., KILICCOTE, H. y KHOSLA, P. K. "Survivable Information Storage Systems", *IEEE Computer*, vol. 33, págs. 61-68, agosto de 2000.

YU, T., HARTMAN, S. y RAEBURN, K. "The Perils of Unauthenticated Encryption: Kerberos Version 4", *Proc NDSS Symposium*, Internet Society, febrero de 2004.

YUVAL, G. "How To Swindle Rabin", *Cryptologia*, vol. 3, págs. 187-190, julio de 1979.

ZACKS, M. "Antiterrorist Legislation Expands Electronic Snooping", *IEEE Internet Computing*, vol. 5, págs. 8-9, noviembre-diciembre de 2001.

ZHANG, Y., BRESLAU, L., PAXSON, V. y SHENKER, S. "On the Characteristics and Origins of Internet Flow Rates", *Proc. SIGCOMM 2002 Conf.*, ACM, págs. 309-322, 2002.

ZHAO, B., LING, H., STRIBLING, J., RHEA, S., JOSEPH, A. y KUBIATOWICZ, J. "Tapestry: A Resilient Global-Scale Overlay for Service Deployment", *IEEE J. on Selected Areas in Commun.*, vol. 22, págs. 41-53, enero de 2004.

ZIMMERMAN, P. R. *The Official PGP User's Guide*, Cambridge, MA: M.I.T. Press, 1995a.

————*PGP: Source Code and Internals*, Cambridge, MA: M.I.T. Press, 1995b.

ZIPF, G.K. *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*, Boston, Addison-Wesley, 1949.

ZIV, J. y LEMPEL, Z. "A Universal Algorithm for Sequential Data Compression", *IEEE Trans. on Information Theory*, vol. IT-3, págs. 337-343, mayo de 1977.

ÍNDICE

D

E

F

G

H

I

J

K

L

M

N

O

P

S

T

U

V

W

X

Z

REDES DE COMPUTADORAS

ANDREW S. TANENBAUM **DAVID J. WETHERALL**

Una introducción clásica y a la vez contemporánea al campo de las redes

Redes de computadoras, 5ª edición, es la introducción ideal al campo de las redes. Este bestseller refleja las tecnologías más recientes sobre este tema con un énfasis especial en las redes inalámbricas, incluyendo 802.11, 802.16, Bluetooth™ y 3G celular, a la par con una cobertura de redes fijas como ADSL, Internet por cable, Gigabit Ethernet, MPLS y las redes de igual a igual. En particular, esta última edición incorpora una nueva cobertura sobre las redes 3G de telefonía móvil, fibra para el hogar, RFID, redes tolerantes al retardo y seguridad en 802.11, además de material adicional sobre enrutamiento en Internet, multidifusión (multicast), control de congestión, calidad del servicio, transporte en tiempo real y distribución de contenido.



Los autores Andrew Tanenbaum y David Wetherall describen los aspectos internos de la red y exploran su funcionalidad, desde el hardware hasta las aplicaciones implicadas, donde sobresalen los siguientes temas:

- La capa física (cobre, fibra óptica, inalámbricos, satélites, OFDM y CDMA).
- La capa de enlace de datos (detección y corrección de errores, ventana corrediza y paquetes sobre SONET).
- La subcapa MAC (Gigabit Ethernet, 802.16, RFID, Ethernet conmutada, redes VLAN).
- La capa de red (algoritmos de enrutamiento, multidifusión, QoS, IPv4, IPv6 y MPLS).
- La capa de transporte (sockets, UDP, TCP, RTP, control de congestión y redes tolerantes al retardo).
- La capa de aplicación (DNS, correo electrónico, Web, medios de flujo continuo, distribución de contenido y redes de igual a igual).
- Seguridad en redes (AES, RSA, IPsec, firewalls, redes VPN, 802.11i y seguridad en Web).

Este libro analiza y describe con detalle los principios asociados con cada capa y después los traduce a través de ejemplos de Internet y las redes inalámbricas.

Para mayor información, consulte la página Web de este libro en:

www.pearsoneducacion.net/tanenbaum


ISBN: 978-607-32-0817-8
90000
9 786073 208178